Dynamic Pricing and Inventory Optimization in E-Commerce using Machine Learning

Louis Etienne Brammer

Dissertation written under the supervision of professor

Pedro Afonso Fernandes

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, 30.12.2025.

Abstract

Dynamic Pricing and Inventory Optimization in E-Commerce using Machine Learning

Louis Etienne Brammer

The current landscape of price and inventory planning has become highly complex and competitive, especially in e-commerce. While simple, linear demand and price forecasting models exist, they fail to capture the subtleties of statistical significance and business policies. Finding simple, stable and interpretable controllers that can operate in a real business context is difficult and costly.
This thesis studies machine-learning driven optimizing of inventory orders and pricing for e-commerce where fluctuating demand as well as capacity constrains and lead times create tightly coupled decisions.
This study develops an operational framework that first learns weekly demand from historical data and then uses that forecast to optimize ordering and pricing under realistic business policies. To do so, several baseline models are trained and benchmarked from which an XGBoost model is selected.
The forecast is then exported and used as the basis for two controllers: a sequential two-step heuristic (first order then price) and an integrated joint controller, selecting price and order simultaneously. The goal of both controller is to optimize and ultimately maximize profits for the forecasted finite horizon while respecting the guardrails (price corridor, service level). The results prove the joint model to be feasible and stable all while delivering superior economic outcomes. This study contributes a deployable, reproducible, interpretable pipeline for joint pricing and inventory control.

Keywords: Time series; Dynamic Pricing; Inventory Optimization; Synthetic Data; Sequential controller; Joint/Integrated controller; Machine learning; Roll-Forward Cross-validation; Combining/ensemble forecasts; Pricing/Inventory Policies; Inventory planning/ordering; Newsvendor Model; E-Commerce.

Resumo

Preços dinâmicos e otimização de inventário no comércio eletrónico usando aprendizado de máquina

Louis Etienne Brammer

O panorama atual do planeamento de preços e stocks tem vindo a tornar-se altamente complexo e competitivo, especialmente no e-commerce. Embora existam modelos simples e lineares de previsão de procura e preços, estes não conseguem captar as subtilezas da significância estatística e das políticas comerciais. Encontrar mecanismos simples, estáveis e interpretáveis que operem num contexto empresarial real é difícil e dispendioso.
Esta tese estuda a otimização, baseada em machine learning, de encomendas e preços para e-commerce, onde a procura flutuante, bem como as restrições de capacidade e os prazos de entrega, criam decisões fortemente interligadas. Este estudo desenvolve uma estrutura operacional que aprende a procura semanal a partir de dados históricos e utiliza essa previsão para otimizar os pedidos e preços no âmbito de políticas comerciais realistas. Para isso, vários modelos de referência são treinados e comparados, a partir dos quais se seleciona um modelo XGBoost.
A previsão é depois exportada e usada como base para dois controladores: uma heurística sequencial, em duas etapas (primeiro o pedido, depois o preço) e um controlador conjunto integrado (selecionando o preço e o pedido simultaneamente).
O objetivo de ambos é otimizar e, em última análise, maximizar os lucros para o horizonte finito previsto, respeitando as diretrizes, como o intervalo admissível de preços e nível de serviço. Os resultados comprovam que o modelo conjunto é viável e estável, proporciona também resultados económicos mais favoráveis. Este estudo contribui com uma pipeline implantável, reproduzível e interpretável para o controlo conjunto de preços e stocks.

Palavras-chave: Séries Cronológicas; Fixação Dinâmica de Preços; Gestão de Stocks; Dados Sintéticos; Controladores Sequenciais; Controladores Integrados; Aprendizagem Automática; Validação Cruzada com Séries Cronológicas; Combinação de Previsões; Modelo do Vendedor de Jornais; Comércio Eletrónico.

Contents

List of Figures

List of Tables

List of Acronyms

ACF Autocorrelation Function.

AI Artificial Intelligence.

AIC Akaike Information Criterion.

ARIMA Autoregressive Integrated Moving Average.

B2C Business to Customer.

CAL Calibration.

CAP Capacity.

COGS Cost of Goods Sold.

CSRD Corporate Sustainability Report Directive.

CV Cross-Validation.

ETS Exponential Smoothing.

FOC First Order Condition.

GDPR General Data Protection Regulation.

GPU Graphics Processing Unit.

KDE Kernel Density Estimation.

KPI Key Performance Indicator.

LOWESS Locally Weighted Scatterplot Smoothing.

MAE Mean Absolute Error.

MAPE Mean Absolute Percentage Error.

ML Machine Learning.

MOQ Minimum Order Quantity.

OTPO Order Then Predict Then Optimize.

PDP Partial Dependence Plot.

RAM Random Access Memory.

RMSE Root Mean Squared Error.

SKU Stock Keeping Unit.

SL Service Level.

UCI University California Irvine.

XGBoost Gradient Boosting Algorithm.

1 Introduction

Since the launch of the Internet and with it, the earliest recordings of online shopping activity in the 1980s, the number of online shoppers has currently reached an all-time high of 2.77bn users compared to less than half of that in 2017 with only 1.09bn users (Lindlahr et al., 2024).
Not only that the predicted amount of online shopping users is expected to exceed 3.5bn by 2029, which represents a penetration rate of almost 50%, meaning that almost every second person on the planet is most likely shopping online (Lindlahr et al., 2024). Translating that into global e-commerce generated revenues the amount did not simply double such as the volume of users but actually more than tripled from 1.507tn (2017) to 4.791tn (2025) and expected to exceed 6.500tn by 2029 (Lindlahr et al., 2024), see appendix 22.
Even though the online segment still generates less than in-store revenues, it is the fastest growing segment, with an expected growth of 12.9% by 2029 (Lindlahr et al., 2024). This fast-growing market presents a unique opportunity for businesses to expand and scale, however it does not come without risks and drawbacks. With the growth of e-Commerce and the increasing purchasing volume, also comes the increasing planning implication, greater complexity of supply chains and logistics. The pressure on supply chains rises (Lindlahr et al., 2024) and it becomes increasingly costly to manage inventory flows in such a fast paced environment (Sun, 2022).
Even though inventory planning is nothing new, with technological advancement and increasing computing power, new statistical methods have been developed to address these issues. In particular the integration of ML methods into business contexts have immensely contributed to forecasting consumer demand, optimize inventory flows and maximizing profitability.
This greatly increasing predictability for businesses toward a constantly growing uncertain market (Ma et al., 2024). With this background a central managerial problem emerges. How can a business translate noisy ML-based demand forecasts into an operational price and inventory controller that respect a strict set of business rules? For the longest time firms often decoupled pricing and inventory control systems despite their economic interdependence. This separation can leave out some yet untapped profits and amplify both stockout and overstocks.
The methodological opportunity lies in bridging forecasting and control mechanisms with policies that are statistically robust and computationally tractable. This thesis addresses this gap by developing and studying two controllers: a sequential baseline and an integrated joint policy both driven by a ML demand model and explicit formulated business rules.

2 Motivation, Objectives and Scope of the Study

2.1 Motivation

Historically, forecasting methods have progressed from approximative rule of thumb methods to modern ML and deep-learning methods. The main goal of those methods never changed: forecast demand dynamics, trends, seasonality and patterns to reduce uncertainty in inventory and pricing planning (Chu et al., 2008).
However, online behavioral shopping patterns differ from the more classic in-store shopping behavior. Search and switching costs are lower, the available information is richer and the competition is only one click away. Consumers might be more sensitive to price changes and have higher price elasticity (Chu et al., 2008). The classical supply/demand problem evolves into a highly complex multilayered problem, making it more difficult for businesses to forecast demand and as a consequence, plan for optimal resource allocation.
A second challenge that emerged is the decision decoupling. Price fluctuations affect demand, which in turn determines service levels and future inventory under certain capacity constraints. However, for the longest time, pricing and inventory have been perceived seperately from each other, sequentially adapting pricing first and in a second step, optimizing inventory. Recent work argues that pricing and inventory are interdependent and have to be considered together and simultaneously in order to reach a true optimal planning policy (Fang et al., 2021).
A third, often under-explored issue, is the forecast-decision gap. Most often predictive models try to minimize statistical loss (e.g. RMSE or MAE), whereas a firm in business tends to optimize for economic objectives such as profit or optimal warehouse logistics.
Following this logic, forecasts that are statistically accurate on average, might yield sub-optimal economic results. Thus, the business policies (e.g. price-corridor, capacity) must be translated into the demand model and controller in order to generate meaningful results both statistically and economically.

2.2 Objectives

The objective of this research is to build and test two reproducible decision frameworks: one sequential model and one joint data-driven forecaster with operational control policies. The two frameworks will be based on a single baseline model. Several baseline models will be tested (ETS, ARIMA, RIDGE, XGBoost) on the online retail dataset and perform demand forecasting. The best performing model will be selected for further evaluation and will serve as the basis that will later be fed into the sequential and unified controllers independently 24 (see appendix).
To keep the business relevant context, the results of both frameworks will be expressed in Key Performance Indicator metrics. In a final step, the performance of both models will be compared with the same objective of maximizing the profit for the given horizon under certain business policies. This study also tries to balance realism with simplicity and

methodological transparency in an attempt to remain reproducible and interpretable.

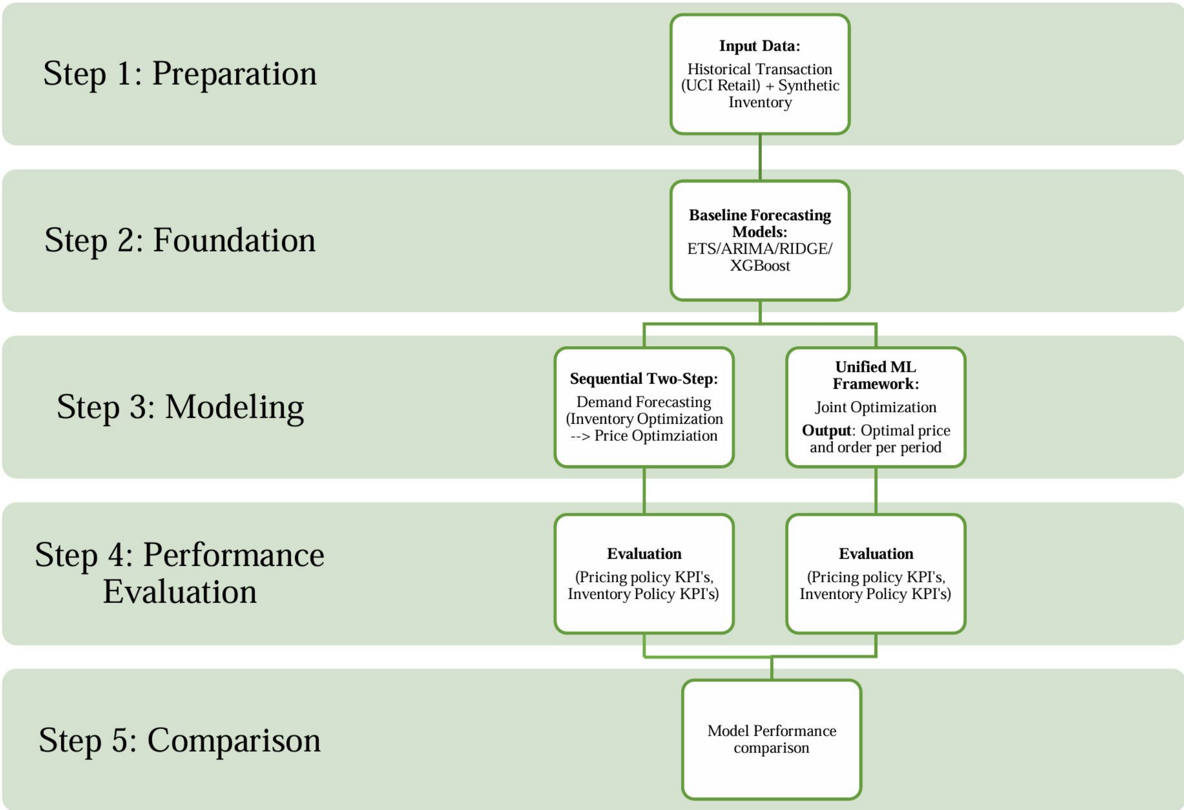

Figure 1: Schematic Overview: Full Methodological Framework

The Schematic figure above highlights the overall flow of this study from top to bottom. The main idea is that of a parallel workflow that forks out and comes back together for evaluation at the end. A more detailed view of the schematic workflow structure can be viewed in the appendix: 25.

2.3 Hypothesis

To answer the problem of profit optimization and maximization. The main hypothesis will be pursued in the thesis

$H1$: A unified ML-Framework that co-optimizes price and inventory yields higher total profit than a sequential approach.

2.4 Scope of the Thesis

This research focuses on developing and evaluating a unified machine learning framework for dynamic pricing and inventory optimization in the online business-to-consumer (B2C) retail sector. The scope is deliberately limited to new goods sold on an anonymized online retail platform for ”all-occasion gift-ware”.

It explicitly excludes business-to-business (B2B) transactions as well as used goods transactions (such as eBay), where pricing dynamics are driven by different mechanisms.
The forecasting and optimization models are designed to operate over a finite planning horizon, reflecting realistic decision-making cycles in retail (weekly planning over several months). The dataset for the research analysis observes only the European and British (UK) e-commerce markets.
Even though other non-European datasets were available, they were harder to interpret since they were keeping track of products that were not inherently typical European and expressed in foreign currency that would be harder to understand the magnitude of.
It is important to note that gift-ware is expected to have strong seasonal effects since consumers tend to ”wrap up” more gifts during holiday season which is usually before and around the Christmas holidays. This is crucial in this study since there is also the goal to forecast model performance when this market experiences demand shocks.

3 Assumptions

In order to develop the models that will be used some fundamental assumptions to frame the problem at hand need to be laid down. These assumptions can be categorized as the following:

3.1 Data and Market Context

In the context of Data and e-commerce market, following assumptions are set: One single distribution channel (no-multichannel strategy) for B2C sales is assumed. The goods are new and non-perishable.
No competition is assumed in the models for several reasons. Collecting price/demand and inventory information from multiple competitors for similar products is simply too costly in terms of resources (time, energy, compute). It is important to note that the demand function is derived from real data which took place in a competitive environment. This way the results and forecasts can still be interpretable in a realistic manner.
In terms of products, assuming a single fulfillment node (one SKU only): the Models will take decisions at a single aggregated SKU fulfillment level (no multi-echelon). To be more precise, the research will filter all other SKU's out to only focus on one SKU. The choice for the product has been based on the most frequently bought product, which simplifies observing seasonal changes and reduces data scarcity.
It is assumed that the single SKU is representative for the majority of the gift-ware products available on the online-store. This way the model keeps its interpretability and simplicity while having enough observations to be statistically significant.
The online-store is assumed to have one single warehouse from where all products are shipped. Due to data unavailability, distance, location and other geographical factors are not taken into account for this research (assuming static, single location conditions).
Non negative quantities or prices are assumed. More precisely, certain transactions are not taken into account such as returns, balance adjustments or other financial flows that might influence pricing outcome. Only sold units are observed. Negative quantities or prices typically represent reverse logistics, which are out of scope for this revenue optimization. Model simplicity and explainability are prioritized over complexity.

3.2 Demand and Pricing (Baseline model)

In terms of demand and pricing policies, following assumptions will set the frame for the models: The consumer Demand $D_t(p_t)$ is a continuous, non-negative and monotonic decreasing function of price p_t (price dictating demand in a standard price/demand fashion). For each time period demand depends only on current price and observable features (no unmodeled cross-period strategic behavior, Markov property). The lead time L (total time from start to finish of a process) is fixed (constant) and known ($L = 2$ weeks) and all orders arrive in full after L. The holding, ordering, and shortage costs denoted as

$$(Ch, Co, Cs) \tag{1}$$

are known and constant over the finite horizon t. Seasonality and trend effects are stationary within the chosen horizon after transformation. The prices and inventory decisions are reviewed weekly (periodic review $T = 1$). The warehouse or supplier has sufficient capacity to fulfill placed orders. An elasticity price stability is assumed, short-horizon price elasticity is stable (or slowly varying) over the forecast horizon.
These assumptions should enable us to set the environment for the models and gain valuable, measurable and interpretable insights. The assumptions also help to reduce the noise from other potential exogenous effects that could influence the model performance.

3.3 Necessity for synthetic inventory Data

Most company inventory data is not publicly available, especially in Europe firms have to comply not only with GDPR but also CSRD regulations. The available data is often also difficult to use for research purposes. Under IFRS regulation, European companies do not have to publicly disclose inventory flow but only have to present certain information such as accounting policies (IFRS, 2016).
Even though the UCI retail dataset at hand carries granular features 23 (see appendix), crucial information relevant to perform inventory optimization including is missing: quantity on hand, location, cost, sales history, supplier information and many other variables. Since dynamic pricing and inventory optimization are based on this type of information this creates a necessity to generate a simple synthetic dataset that presents those features and reflects the transactions from the original UCI dataset in a meaningful way. To do that, some initial assumptions need to be defined in order to generate synthetic inventory data.

Inventory System (Synthetic generator)
The synthetic inventory system is designed to follow a classical reorder-point structure, serving both as the basis for data generation and as the control mechanism within the baseline and unified models. The setup ensures internal consistency and interpretability while maintaining computational tractability.

Policy structure Inventory dynamics are simulated using a standard reorder-point policy, denoted as (s, Q). Whenever the inventory position (on-hand + on-order) falls at or below the reorder point s, a replenishment order of fixed size Q is placed. This policy provides a simple yet realistic representation of many retail replenishment systems and serves as the baseline control in both frameworks.

Initial stock The initial inventory level I_0 is determined based on average demand during lead time multiplied by a service factor (typically between 1 and 2) ensuring sufficient safety coverage:

$$I_0 = k \cdot \bar{D}_L, \quad k \in [1, 2], \tag{2}$$

where $\bar{D}_L$ is the expected demand during lead time L.

Fixed lead time Lead time L is assumed to be fixed and known in advance, representing the number of periods between order placement and receipt. In this study, L is set to represent a short, realistic e-commerce lead time (here $L = 14$ days).

Safety stock Safety stock (SS) is computed according to the target service level using the standard normal safety factor z:

$$SS = z \cdot \sigma_{LTD}, \tag{3}$$

where σ_{LTD} is the standard deviation of demand during lead time and z corresponds to the chosen service level (e.g. $z = 1.645$ for 95%).

Inventory balance Inventory levels evolve dynamically according to realized or forecasted demand. At each period t, on-hand inventory decreases by the minimum of available stock and predicted demand and is not backordered.:

$$I_{t+1} = I_t + Q_t - \min(I_t, \hat{D}_t), \tag{4}$$

where $\hat{D}_t$ is the predicted demand at time t. Unfulfilled demand results in lost sales:

$$L_t = \max(0, \hat{D}_t - I_t), \tag{5}$$

Order receipts Orders placed at period t are received in full after a fixed lead time L with no partial or delayed shipments.

Capacity and batching The system assumes a maximum warehouse capacity of 2.000 units (for simulation purposes) and no supplier capacity constraints allowing full fulfillment of all orders within the lead time. Unless explicitly stated otherwise, there are no batch-size or MOQ restrictions. Replenishment is treated as continuous and may be rounded to the nearest whole unit when simulated.

Rationale These assumptions simplify the optimization process while preserving the essential operational behavior of a realistic e-commerce fulfillment system. By controlling key factors such as lead time, service level and order quantity deterministically the synthetic inventory generator provides a stable foundation for comparing the sequential heuristic and unified ML frameworks under consistent conditions.

4 Literature review

The idea of dynamic pricing as a concept is not novel. The term was first coined by Antoine-Augustin Cournot in 1838, who provided one of the earliest mathematical models describing the relationship between price and demand and addressed the problem of determining the optimal selling price of goods (den Boer, 2015).

Building on this foundation, subsequent work in the 20th century began to examine iterative and adaptive price adjustment schemes. A notable contribution is the work of Aoki (1974), who compared stochastic approximation methods with Bayesian approaches to price setting under certain demand. His work demonstrated that both methods converge asymptotically to equilibrium prices, and importantly, showed how inventory considerations through a terminal stock target, can directly influence price adjustment dynamics. This early integration of stock level into pricing models provides a conceptual link to the more recent literature on joint pricing-inventory optimization, which is central to this thesis.

Since the late 20th century research in dynamic pricing has expanded considerably, particularly with the advent of digitization and increased computational power. Given the breadth of this literature, the present review focuses on more contemporary contributions, specifically those situated at the intersection of machine learning (ML) and e-commerce.

Recent work has demonstrated the potential of ML-driven dynamic pricing systems to outperform traditional static models in terms of profitability and adaptability. For instance, Nowak and Paw lowska-Nowak (2024), Gupta (2025) and Gupta and Jain (2024) investigate cloud-based analytics for scalable dynamic pricing solutions while R. et al. (2025) focuses on profit maximization.

Liu et al. (2020) employ a Bayesian approach to model dynamic pricing under uncertainty and Ma et al. (2024) evaluate forecasting performance using time-series models such as ARIMA, LASSO, Ridge and Elastic Net. Despite methodological differences these studies share a common objective: designing models that improve profit performance and operational efficiency when compared to classical static approaches.

Because this thesis emphasizes forecasting as the foundation of pricing and inventory decisions, it is necessary to also consider literature on ML-based forecasting models. Here, approaches such as reinforcement learning (Huber and Stuckenschmidt, 2020), Markov decision processes and broader AI-based methods (Kagalwala et al., 2025) have been explored for their ability to incorporate uncertainty and adapt to dynamic environments. However, this research is not designed to explore reinforcement learning methods such as bandit or multi-armed bandit methods and will this not be used in this study.

A more recent and emerging research stream is more concerned with the integration of pricing and inventory optimization within a single forecasting and decision-making framework. Hasiloglu-Ciftciler and Kaya (2025) demonstrated the value of such integration for perishable goods (using finite rolling horizon models) showing that joint models not only increase profits but also reduce waste.

New methodological developments include the OTPO paradigm introduced by Zhang et al. (2025), which aims to align forecasting more directly with operational objectives. On the inventory side, Kumar et al. (2024) provide valuable insights into demand forecasting with auto-correlated stochastic (random) processes using exponential smoothing methods.

One major practical limitation of this thesis is the absence of publicly available inventory data. To address this gap Lu et al. (2025) provide a comprehensive review of synthetic data to generation techniques while Chan et al. (2022) applies discrete-event simulation to generate synthetic manufacturing data. Other relevant contributions include Long et al. (2025), who discusses leveraging synthetic data to overcome ML challenges. Studies by Sun (2022) highlight the applicability of synthetic datasets across various applied ML contexts justifying the use of synthetic data for this thesis.

The relationship between dynamic pricing and inventory optimization has been explored in the early works of Thomas (1970) showing that when a monopolistic actor faces deterministic, price-responsive demand over a finite rolling horizon, prices as well as inventory must be chosen simultaneously. This set-up yields significantly better results and an efficient forward algorithm.

Whittemore and Saunders (1977) researched on the same topic, under stochastic demand with multiple replenishment modes, establishing optimal base-stock-type structures.

Thomas (1970) and their research concluded that jointly shaped models create the optimal decision. The way price has been treated as an operational lever in this thesis has been derived from the approach from Bitran and Caldentey (2003). Capacity constrained resources and optimal prices (and maximum profit) depend on remaining inventory and time, further validating the approach that inventory and price are endogenously linked.

The scope of this research is deliberately circumscribed. The focus lies exclusively on price as the control variable and inventory quantities at an aggregated single SKU-level. This measure has been taken to deal with the curse of dimensionality (Banks and Fienberg, 2003). Broader theoretical frameworks such as auction theory (Segal, 2003), mechanism design or sector-specific applications, telecommunications, road/fuel pricing (de Palma et al., 2006), electricity and weather dependent pricing are explicitly excluded. They fall outside the retail-focused objectives of this thesis. Furthermore the forecasting will be limited to a finite rolling horizon within a specific time frame (Kincaid and Darling, 1963). This finite horizon will be defined in the following section with respects to the heuristic problem at hand (Gallego and van Ryzin, 1994).

5 Methodologies, methods and tools

5.1 Methodologies Overview

The methodology of this quantitative research will be segmented into five main parts that represent a systematic approach to the dynamic pricing and inventory optimization problem. First of all, the historic dataset itself will be discussed as well as the pre-processing methods applied to it in order to run the models. More specifically, the approach adopted to create the necessary synthetic inventory dataset will be presented. Then, the focus will deepen into the forecasting model choices that will later be used, explaining why respective models have been chosen for each of the sequential and unified frameworks. The third step will be a look into the sequential model itself, how it is structured and composed, followed by an explanation of the unified ML framework, which is the centerpiece of this research.
A more in depth look at the unified framework will reveal its components, main objective, relevant inputs, optimization routine and model evaluation. The technical means (such as computer, GPU, RAM and tools) that were employed will be discussed, for more reproducibility.
Finally, it is important to mention that this thesis is simulation-based. The utilized data is composed by historical and synthetic data which aim is to recreate a realistic scenario on how prices, customer demand and inventory stock flow behave. For the sake of balancing interpretability and performance the amount and complexity of the features used in the models have been kept to a minimum.

5.2 Data and Pre-processing

In this section the rationale behind why the dataset has been chosen for this research will be explored. The dataset ”Online Retail II UCI” is an e-commerce retail shop dataset containing all the transactions for a UK-based, purely online, retail shop. The dataset holds 1.067.371 instances, spans from 01/12/2009 until 09/12/2011 and has been gathered by D.Chen (2012) as part of a research project (Data Mining for the retail industry[...]) within the UCI.
The dataset is particularly valuable for research with associated tasks such as classification, clustering and regression since it is composed of different granular data types such as Multivariate, Sequential, Time-Series 23 and Text for a longer period of time (two years). The dataset showed almost no missingness making it very fit for analyses by reducing data cleaning and assumptions.

Name	Type	Description
InvoiceNo	**STR**	Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
StockCode	**STR**	Product (item) code. Nominal. A 5-digit integral number uniquely assigned to each distinct product.
Quantity	**INT**	The quantities of each product (item) per transaction. Numeric.
InvoiceDate	**DATETIME**	Invoice date and time. Numeric. The day and time when a transaction was generated.
UnitPrice	**FLOAT**	Unit price. Numeric. Product price per unit in sterling (£).
CustomerID	**STR**	Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.
Country	**STR**	Country name. Nominal. The name of the country where a customer resides.
Description	**STR**	Product (item) name. Nominal.

Figure 2: Online Retail II UCI Dataset (D. Chen, 2012) Non-altered

As introduced earlier, there have been some difficulties in gathering a dataset containing both granular SKU information over a longer period of time as well as the inventory flow data. Due to that lack of availability, a deliberate decision has been taken to generate a synthetic inventory dataset based on the demand patterns of the historical dataset.

What is important to add is that the inventory only focuses on the inventory flow of one single SKU to keep the training data from exploding. All the relevant inventory features that have been created in order to generate the synthetic dataset can be seen in the annex 27, 28 and 29.

The model assumes a positive lead time L of 2 weeks for more realism (a lead-time of 0 = immediate replenishment) (Hasiloglu-Ciftciler and Kaya, 2025). This represents a short but realistic replenishment delay for non-complex, non-customized retail products typically distributed within Europe.

Since the goods in scope are standard consumer items longer lead times are unnecessary and would overstate supply-chain uncertainty. A weekly review period ($T = 1$) was selected to reflect standard retail planning cycles allowing regular adjustments to stock and price while avoiding excessive operational noise. The eight-week rolling average for demand captures medium-term seasonality and smooths transient fluctuations providing a stable basis for reorder-point estimation (Kumar et al., 2024). A minimum of four weeks of data was required before rolling statistics were computed to ensure reliable initial estimates and reduce bias from sparsely observed early periods.

The chosen service level of 90% aligns with common retail benchmarks balancing inventory cost and stockout risk. Other control parameters such as the use of weighted prices (revenue-weighted averages) and the (S, T) inventory policy were selected for their interpretability and consistency with established inventory management practices in single-product retail settings. This set up allows for realistic inventory stock simulations (Lu et al., 2025).

5.3 Forecasting Models

The forecasting models in this thesis are designed in such a way that the forecasting layers are prioritizing models that can balance transparency and operational robustness as well as predictive strength. These choices should enable to use the findings for managerial reporting rather than chasing marginal improvements from opaque models.

In a first instance, a more classical time-series model for baseline modelling will be applied. This model includes ETS capturing seasonality with few parameters. It is stable on small samples and can yield interpretable components. It might serve as a baseline model that is variance adapted, and performs well especially when demand is noisy. However, it might be limited (James et al., 2023).

An ARIMA model will provide an interpretable stochastic benchmark and be useful to test if feature-based ML adds a true value beyond seasonality and autocorrelation (James et al., 2023). Those Baselines will help to set an accuracy floor as well as help isolate the value of price features and nonlinearity.

In a second instance time-based machine learning will be implemented using a tree based model, for nonlinear price-demand effects and seasonality capture. (Breiman, 2001).

XGBoost will provide a stronger tabular performance with more price sensitivity, regularization and monotonicity constraints (Chen and Chen, 2015).

The datasets present a multitude of features that could potentially be correlated (multicollinearity). The Ridge regression model simplifies the complexity of the model by mitigating the impact of each feature without taking them out of the model. In scenarios such as control or optimization, dropping factors (to counter correlation) might result in unsatisfactory outcomes. Thus, Ridge is the adequate method for the problem at hand (Hoerl and Kennard, 1970).

Appropriate evaluation metrics will be employed to verify not only if the prediction is off but also by how much and if the model is systematically under- or overpredicting. The following metrics will be chosen: MAE, RMSE and MAPE. MAE has been chosen since it indicates by how far off the forecast is (in terms of the model, here currency in Pounds £).

Even though the MAE is simple, it fails to indicate the direction of the error (positive/negative). RMSE has been chosen for its qualities to detect larger error that are not caught by MAE.

Finally, MAPE will be used to check the absolute error as a percentage of the actual values. This is particularly useful when wanting to compare different forecast (which is the case here). MAPE is particularly interesting when wanting to evaluate performance in time-series scenarios with sudden demand jumps.

5.4 Two-Step Heuristic (Sequential Approach)

After having set up the baseline model, the two-step heuristic pipeline follows in which pricing and inventory control are optimized independently but sequentially linked through demand forecasts. During the first stage demand forecasting is performed using the historical sales and synthetic inventory dataset incorporating features such as past demand, prices, and temporal variables.

The best performing forecasting model will estimate the expected demand over a defined finite horizon, capturing how changes in price influence future sales volumes. During the second stage the pricing policy is adjusted dynamically within that forecasting horizon to maximize revenue or profit based on the estimated demand/price elasticity. Once prices are determined, the inventory control component applies a replenishment policy specifically, a periodic review (S,T) or reorder point (s,Q) model, to determine optimal order quantities.

This inventory decision integrates both the updated demand forecasts and adjusted pricing, ensuring alignment between sales expectations and stock availability. The sequential nature of this heuristic allows demand prediction, pricing and inventory management to be logically connected, yet computationally separated providing a benchmark for evaluating the unified ML framework.

5.5 Unified ML Framework

In parallel to the sequential two-step heuristic the unified machine learning framework integrates demand forecasting, dynamic pricing as well as inventory optimization into one single, simultaneous decision-making process. The objective of this joint policy model is to determine both the optimal selling price and replenishment quantity concurrently, such that expected profit is maximized across the forecasting horizon. This method follows the rationale that price and inventory cannot be planned and perceived independently from each other (Federgruen and Heching, 1999).

Similar to the sequential model, the joint model uses the best performing model as a basleine. The model then selects input features consisting of price, historical demand patterns, inventory states, and time-related variables, thereby ensuring that pricing and inventory decisions are mutually informed.

The optimization routine then selects the price p_t and order quantity Q_t for each period t maximizing expected profits, subject to operational constraints such as holding-, ordering- and shortage costs. This integrated formulation ensures that inventory availability directly influences pricing decisions while anticipated demand simultaneously drives replenishment quantities, effectively integrating both control dimensions. Model evaluation follows the same procedure as in the baseline but focuses on profitability, service level, and operational efficiency gains relative to the sequential approach.

The following Equation shows the main goal of the joint approach profit maximization:

$$
\begin{aligned}
\max_{p_t,\,Q_t} \quad & \mathbb{E}\left[\pi_t\right] = \mathbb{E}\Big[\, p_t\,\hat{D}_t(p_t, I_t, X_t) - C_h\, I_t - C_o\, Q_t - C_s\, L_t \,\Big] \\
\text{s.t.} \quad & I_{t+1} = I_t + Q_t - \min\big(I_t,\ \hat{D}_t(p_t, I_t, X_t)\big), \\
& I_t \ge 0, \quad Q_t \ge 0, \\
& L_t = \max\big(0,\ \hat{D}_t(p_t, I_t, X_t) - I_t\big), \\
& Q_t \text{ arrives after a lead time } L.
\end{aligned}
\tag{6}
$$

This equation will later serve as a basis to find the highest and above all most optimal price within the allowed price boundaries.

6 Synthetic Inventory Generation

6.1 Exploratory Data Analysis

Publicly availabe retail datasets rarely include operational inventory states (on-hand, pipeline, stockouts). To evaluate pricing–inventory policies, a simulated and operationally consistent SKU-week inventory trajectory driven by observed sales and prices from the UCI Online Retail II data is necessary. A classical reorder-point policy (s, Q) with a fixed lead time L and service level target (based on z) is adopted.
The synthetic inventory is generated from the historic retail data and follows the rationale of Goltsos et al. (2022).

Inputs, engine, ouputs:

- **Input:** Weekly demand (from UCI), price, lead time L, service level α, (s, Q) rules.
- **Engine:** Inventory balance, lost-sales, pipeline with delay L.
- **Output:** on_hand_begin, orders_placed, arrivals, on_hand_end, lost_sales, service level.

Data layout and behavior:

- **Granularity:** One high-volume SKUat weekly frequency.
- **Core columns:** WeekStart, price, demand_real, avg_weekly_demand, reorder_point_s, order_up_to_S, on_hand_begin, orders_placed, arrivals, sales_fulfilled, lost_sales, on_hand_end, inventory_position, stockout_flag, lead_time_weeks.
- **Behavior:** Inventory follows a sawtooth pattern (depletes with demand, jumps on order arrival after L weeks). Reorder decisions occur when inventory_position $\leq s$.

A snippet of the final synthetic weekly SKU dynamics and demand can be reviewed in the appendix: (26, 30).
The estimated rolling demand statistics (Huber and Stuckenschmidt, 2020) is followed, to capture local demand and enable better smoothing with safe fallbacks for early weeks:

$$\bar{D}_t = \frac{1}{W} \sum_{i=t-W+1}^{t} D_i, \quad W = 8 \tag{7}$$

$$\sigma_t = \sqrt{\frac{1}{W-1} \sum_{i=t-W+1}^{t} (D_i - \bar{D}_t)^2} \tag{8}$$

Set policy parameters (time-varying):

$$s_t = \mu_L + z\,\sigma_L, \qquad S_t = \mu_{L+T} + z\,\sigma_{L+T} \tag{9}$$

where

$$\mu_L = L \cdot \bar{D}_t, \qquad \sigma_L = \sigma_t \sqrt{L}, \tag{10}$$

and analogously for $(L + T)$:

$$\mu_{L+T} = (L + T) \cdot \bar{D}_t, \qquad \sigma_{L+T} = \sigma_t \sqrt{L + T}. \tag{11}$$

Initialize:

$$I_0 = \max\left(S_0,\ k \cdot \bar{D}_L\right), \qquad k \in [1, 2] \tag{12}$$

where I_0 is the initial inventory level, S_0 is the initial order-up-to level, and $\bar{D}_L$ represents the expected demand during lead time L.
Simulate weekly (lost-sales):

Arrivals: add pipeline delivery after exactly L weeks.
Sales: sales_fulfilled = min(on_hand_begin, demand_real); lost_sales = demand_real - sales_fulfilled.
Update Inventory:

$$I_{t+1} = I_t + \text{arrivals} - \text{sales\_fulfilled} \tag{13}$$

Reorder rule: if inventory_position $\leq s_t$, place order of size Q (or $inv\_pos\ S_t\ inv\_pos$ for $S\ policy$).
The histogram [30] (see appendix) displays the distribution of weekly demand for the selected SKU in the synthetic dataset. It exhibits a slight right-skew, typical of retails sales, reflecting occasional high-demand spikes. The mean is ≈913 units and the media ≈831 units, confirming a longer upper tail.
As shown in the ACF [31] (see appendix), only a few lags lie outside the blue confidence band, indicating limited statistically significant autocorrelation. Modest correlations of about 0.24–0.26 at lags 1 and 6 suggest short-term inertia (e.g. repeat purchases or slowly adjusting demand). The fast decay of the ACF implies a largely stochastic series rather than an overly smoothed pattern,desirable for a synthetic demand process meant to mimic realistic short-horizon behavior.

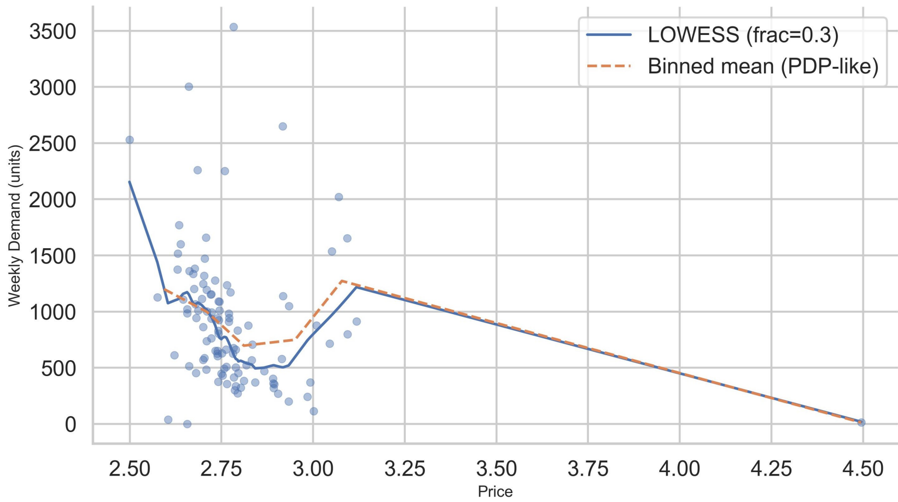

Figure 3: Price vs demand_real with LOWESS/PDP style curve to visualize price effect.

The scatterplot with a LOWESS fit and binned mean (PDP-like) trend shows demand massed around £2.75. Above this level demand falls sharply and approaches zero at high prices consistent with a downward-sloping demand curve. The fitted lines indicate non-linear elasticity: small changes near the central band move quantity modestly whereas larger increases cause disproportionate drops. A slight bump near £3.00 suggests local heterogeneity or transient effects (e.g. promotions or stock conditions). Overall, the pattern is smooth and interpretable, supporting the realism of the synthetic price–demand relationship for later forecasting and optimization.

Inventory-related distributions on_hand_begin and on_hand_end should concentrate between s_t and S_t if tuned well. Recommended plots:

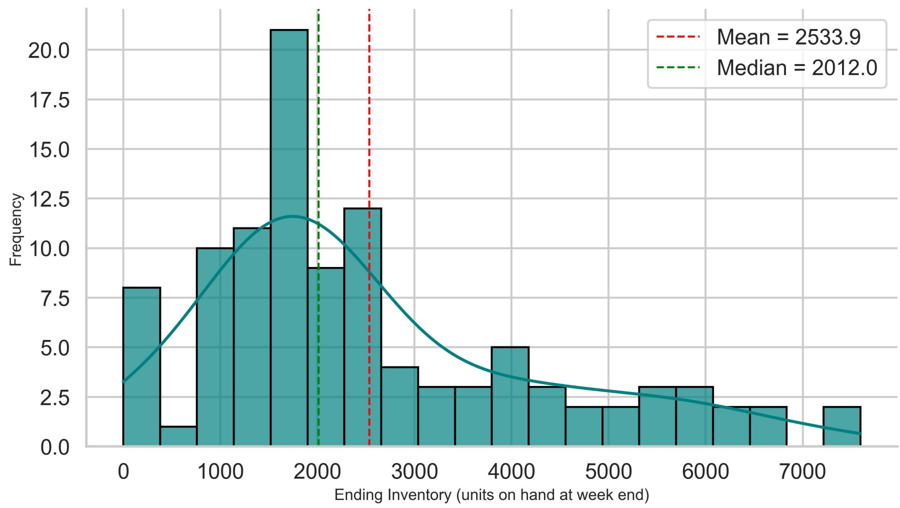

Figure 4: Histogram of on_hand_end

The histogram above shows the distribution of ending inventory levels on_hand_end which exhibits a right-skewed shape typical of replenishment systems since inventory accumulates after restocking and gradually depletes as demand consumes the stock. The concentration of values around 2.500 units (weekly mean close to the reorder threshold for the simulated horizon) indicates a stable and well-regulated replenishment cycle, while the long upper tail reflects occasional overstock periods following replenishment arrivals or lead-time delays. The histogram of orders placed [32] (see appendix) further supports this behavior by illustrating the zero-inflated distribution of weekly orders placed. Most weeks register no replenishment activity, as inventory remains above the reorder point while infrequent but large orders appear as spikes corresponding to restocking events (max. order: ≥ 5.000 units).

Time series structure

Sawtooth inventory levels with arrivals lagged by L: reorders occur when inventory_position crosses s_t. If the variable: avg_weekly_demand follows seasonal patterns, s_t and S_t should co-move (higher in peak months). Stockouts (if any) align with demand spikes and late arrivals.

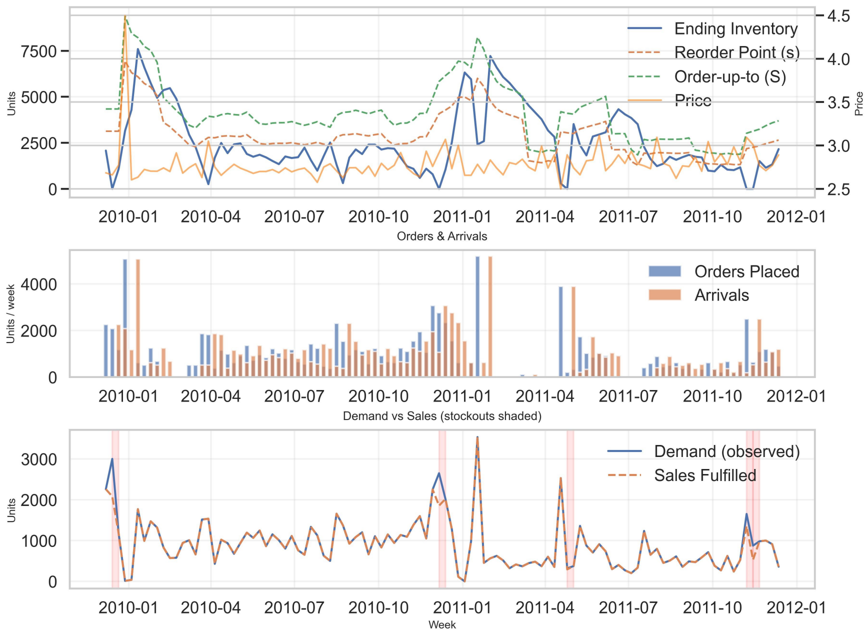

Figure 5: Time series multi-axis

Top panel: Top: Ending inventory tracks the *(s, S)* policy whenever on-hand falls to the reorder point s an order lifts inventory toward S, producing the visible spikes.

Middle panel: Bars show orders placed and two-week-lagged arrivals. Quiet periods (few orders) indicate inventory comfortably above s. Clustered orders occur around demand surges, notably early-2010 and around 2011-01.

Bottom panel: Observed demand and fulfilled sales largely overlap Red bands mark brief stockouts (demand exceeds available units before arrivals). Overall, the policy maintains service levels with only short, transient shortages.

The Figure 35, (see appendix) depicts that the demand mean is broadly stable with episodic peaks; volatility (std. dev.) rises during those peaks and subsides thereafter.

6.2 Synthetic Inventory Results

The synthetic inventory generator behaves as expected under the fixed reorder-point policy (s, Q), producing seemingly realistic inventory dynamics over time. Using the historical transaction data from the UCI Online Retail II dataset as a foundation, the generator successfully reconstructed weekly demand patterns, price–demand relationships, and inventory dynamics in line with classical reorder-point policies s. The results across Figures 3–5; 30–35 demonstrate that the system captures the essential behavioral and stochastic characteristics of real-world retail operations.

The demand distribution [30] (see appendix) follows a right-skewed pattern, indicating natural variability and occasional demand spikes. The autocorrelation plot [31] (see appendix) confirms limited serial dependence, reflecting a realistic degree of short-term persistence without artificial periodicity. The price–demand relationship [3] exhibits the expected negative elasticity, validating the economic plausibility of the generated data.

Finally, the rolling mean and standard deviation analysis [35] (see appendix) reveals adaptive nonstationary demand with controllable volatility. That is precisely the type of variation required for dynamic pricing and inventory optimization experiments.

These results are aligned prior research emphasizing the necessity of synthetic data when real operational logs are unavailable (Lu et al., 2025; Chan et al., 2022; Long et al., 2025; Sun, 2022). The approach follows best practices outlined by Lu et al. (2025), ensuring representativeness through statistically grounded transformations, while Chan et al. (2022) and Long et al. (2025) similarly highlight the importance of simulation-driven synthetic datasets in validating ML-based decision models.

7 Baseline Forecasting

7.1 Objective

This section develops a step-by-step procedure to build a robust baseline model for demand forecasting. The main objective is to systematically compare different linear and non-linear approaches, specifically: Exponential Smoothing ETS, Autoregressive Integrated Moving Average ARIMA, Ridge Regression, and XGBoost, in order to identify the model that delivers the highest predictive accuracy over the relevant forecast horizon. Furthermore, beyond benchmarking, the purpose of this section is to demonstrate that at least one of these relatively standard, well-understood models can produce forecasts of sufficient quality to support downstream decision-making.

The resulting baseline (demand) forecast will be used as a key input to the subsequent inventory optimization module and the dynamic pricing engine, providing a clear reference point against which more advanced sequential and unified frameworks can be evaluated.

7.2 Data Inputs

The figure below summarizes the data inputs and feature sets used for each of the four baseline models in chronological order. The listed exogenous features and rationale should provide a glimpse into the models approach and what features where used to perform the demand forecast over the defined forecast horizon.

The table 34 (see appendix) depicts the deliberate progression in the richness and complexity of information available to each baseline model. ETS and ARIMA operate on a strict univariate weekly demand series and therefore quantify how much structure can be extracted from the historical trajectory alone.

Ridge and XGBoost, in contrast, exploit increasingly elaborate sets of lagged, smoothed and calendar-based features, and in the case of Ridge also operational and price-related covariates. This design makes it possible to assess not only which algorithm performs best but also how much incremental accuracy is obtained by moving from simple time-series models to feature-rich linear and non-linear approaches.

7.3 Models Tested

The first method to be tested is exponential smoothing methods (ETS) which is simple and widely used for time-series forecasting and can often perform competitively with more complex models (Kumar et al., 2024). In this study, an additive ETS specification was fitted to the weekly demand series using the training horizon and then extrapolated over the test horizon (see figure below). The resulting forecast, shown as the green dashed line, degenerates into an almost linear downward trajectory and fails to reproduce the variability, local peaks and patterns observed in the actual test demand. In particular, the model does not capture any seasonal pattern or short-term trend reversals and systematically underestimates demand over most of the forecast horizon. This behaviour indicates that, for the present dataset, a simple univariate ETS baseline is unable to exploit the underlying demand dynamics and provides only a weak benchmark against which the other models can be compared.

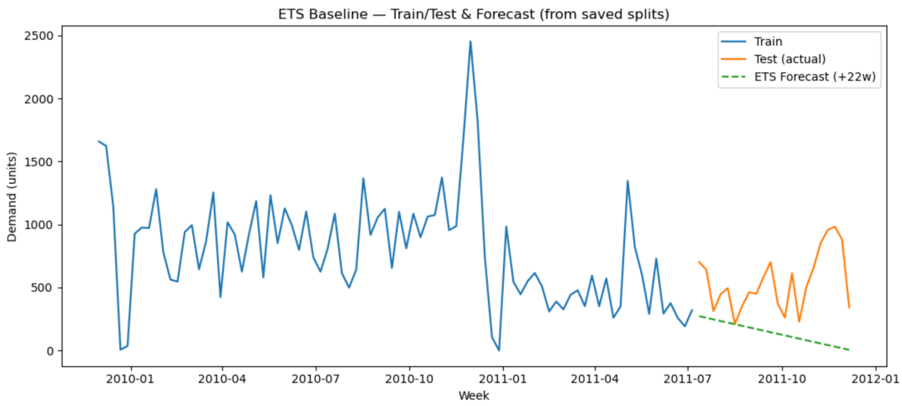

Figure 6: ETS Baseline Forecast (Model fit over test data holdout)

The next method, Autoregressive Integrated Moving Average (ARIMA), is most often used to better capture linear temporal dependencies and seasonality in sales time series as shown by (Ma et al., 2024). In this experiment, an ARIMA (0,1,2) specification was selected on the basis of the AIC and fitted to the weekly demand series, with forecasts for the test horizon plotted as the green dashed line in the figure below.
Similar to the ETS baseline, the ARIMA forecast flattens around a slowly declining mean level and fails to reproduce the pronounced fluctuations observed in the actual test demand. The model does not anticipate any local peaks and patterns in the held-out data and tends to systematically underpredict demand, especially towards the end of the horizon (highest peak in real test dataset). Although ARIMA provides a more flexible univariate benchmark than ETS, its performance on this dataset remains limited and indicates that richer model classes are required to exploit the observed demand dynamics.

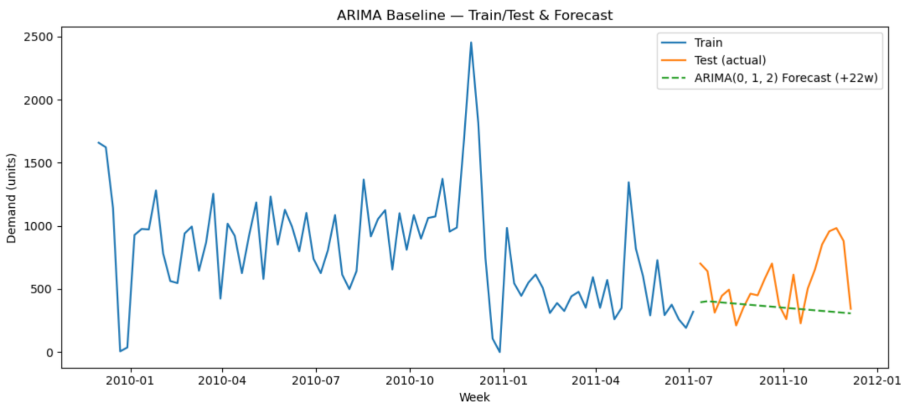

Figure 7: ARIMA Baseline Forecast (Model fit over test data holdout)

Ridge Regression for regularization has been used here to control for multiple features without deleting them (Ma et al., 2024). As pointed out in (Hoerl and Kennard, 1970) foundational work, it is included for its robustness to multicollinearity and stable coefficient estimation under correlated predictors.
The resulting ridge forecast on the test horizon (green dashed line in the figure below) tracks the general level of demand more closely than ETS and ARIMA: it lightly adjusts upwards over time and remains in the correct magnitude range. However, the forecast is relatively smooth and fails to reproduce the sharp local peaks and patterns observed in the actual test demand, systematically underestimating the largest spike and overpredicting some of the low-demand weeks. Thus, while the multivariate ridge model represents an improvement over purely univariate baselines (ETS/ARIMA), it still struggles to capture the full amplitude and variability of the true demand dynamics.

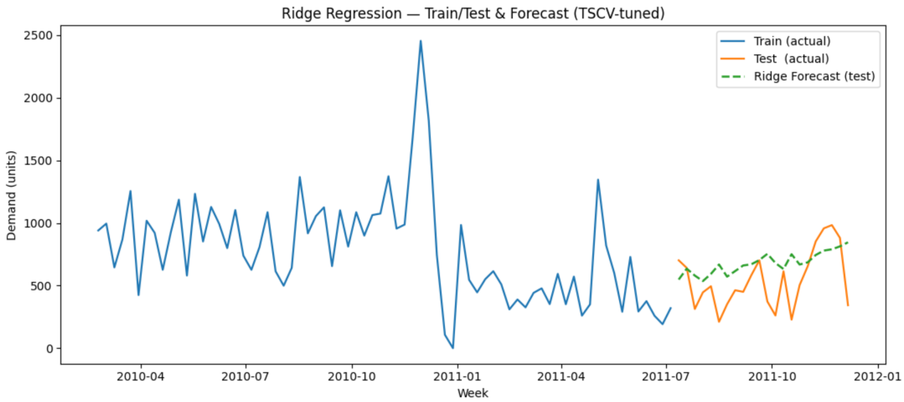

Figure 8: Ridge Regression Baseline Forecast (Model fit over test data holdout)

A tree based regressor method such as XGBoost might be more adapted. In this case it could hold the ability to capture non-linear and complex interaction (Wang et al., 2024). Especially in the context of dynamic pricing, gradient boosting methods have shown to perform well (Ma et al., 2024).

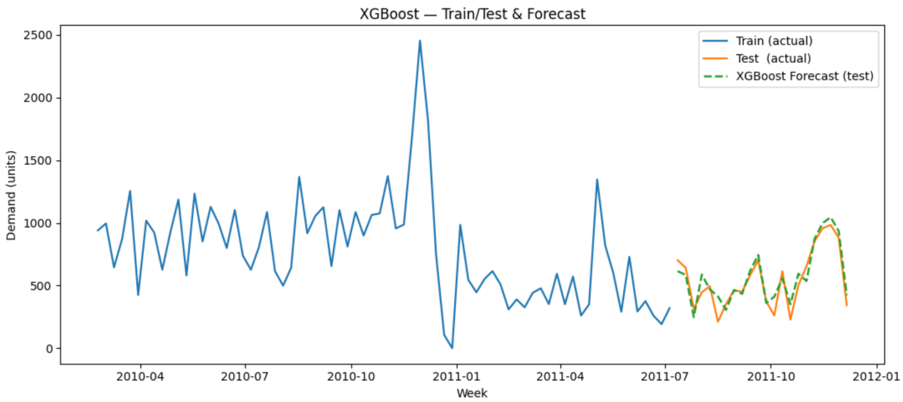

Figure 9: XGBoost Baseline Forecast (Model fit over test data holdout)

In the present experiment, the XGBoost model is trained on an extended set of lagged and rolling demand features together with seasonal calendar encodings. The forecast on the test horizon is again shown as a green dashed line (see figure below).

In contrast to the previous baselines, the XGBoost predictions (see figure 9) closely track both the level and shape of the realized demand, reproducing the overall upward trend and aligning well with the local peaks and patterns. While small deviations remain around some extreme observations, the forecast envelope largely overlaps with the actual series, indicating that the non-linear boosted trees are able to exploit a richer portion of the underlying demand structure than the linear and purely univariate models.

7.4 Results

Since the plotted forecast on a holdout test set alone is not sufficient to decide which baseline model should be carried forward, quantitative error metrics are used to make a transparent, data-driven comparison. In this context, three complementary measures are reported for each model on the test horizon: MAE, RMSE and MAPE.

7.4.1 Evaluation Metrics

The evaluation metrics used will be:

- **MAE:** Summarizes the average (mean) absolute deviation between forecast and realized demand. It is easy to interpret in the natural unit of the series (units sold in Pounds £) and is relatively robust to occasional large errors. Making it useful for assessing the typical magnitude of forecast errors that would directly translate into systematic over- or under-stocking.
- **RMSE:** Is more sensitive to large errors because the deviations are squared before averaging. In an inventory and pricing setting, occasional large forecast errors can be particularly costly (e.g. severe stockouts or large overstocks), so RMSE helps to highlight models that avoid such extreme mistakes even if their average error is similar.
- **MAPE:** Expresses the average absolute error as a percentage of actual demand. Despite its limitations at very low demand levels, it is informative in this case for judging whether a model's errors are acceptable relative to the size of the underlying demand.

7.4.2 Model Error Metrics

The table below highlights the different evaluation metric results and performances. ETS performs worst across all metrics (MAE $\approx$ 407 units, RMSE $\approx$ 488 units, MAPE $\approx$ 67%), confirming the visual impression that its forecasts fail to track the level and variability of demand. ARIMA reduces the error substantially (MAE $\approx$ 233, RMSE $\approx$ 306, MAPE $\approx$ 38%), but still leaves large deviations in both absolute and percentage terms. Ridge Regression achieves slightly lower MAE and RMSE than ARIMA (MAE $\approx$ 197, RMSE ≈ 253), indicating an improvement in typical and squared error, but its MAPE remains relatively high (≈ 57%), reflecting poor relative accuracy in some low-demand weeks.

Model	MAE	RMSE	MAPE
ETS	407,25	487,84	67,28%
ARIMA	233,19	305,74	38,19%
RIDGE	196,87	252,73	57,27%
XGBoost	79,31	88,30	19,86%

Figure 10: Evaluation Metric Performance (Summary)

XGBoost clearly dominates the other approaches on all three metrics, with MAE ≈ 79 units, RMSE ≈ 88 units and MAPE below 20%. This indicates that the non-linear, feature-rich gradient boosting model is able to capture the key patterns in the demand series while keeping both typical and large errors at a comparatively low level. Given this consistent robustness and the fact that the resulting error magnitudes are acceptable for a proof-of-concept setting, XGBoost is selected as the baseline forecasting model for the subsequent inventory optimization and dynamic pricing experiments.

7.5 Final Selection and Forecast

To ensure that XGBoost is a stable defensible baseline demand forecaster that can later be fed to the sequential controller and unified optimization, it is important to make sure that it does not exhibit temporal leakage.
To ensure that, evaluation using time-series cross-validation (rolling/forward splits) can be applied. It preserves chronological order in every fold and within each fold, the training segment is further split to enable early stopping. The test segment in each fold simulates an unseen future slice. The advantage of this procedure is that it mirrors how the model will be used in production: fit on the past, predict the immediate future and roll forward.
Across folds, the model tracks level and short-term movements reasonably well with predictions closely following actuals in most weeks. The main systematic miss is peak attenuation 38, 39 (see appendix) during surge weeks the forecast underestimates amplitude (see figure below) and sometimes lags the upswing by one step, while troughs are captured more cleanly.

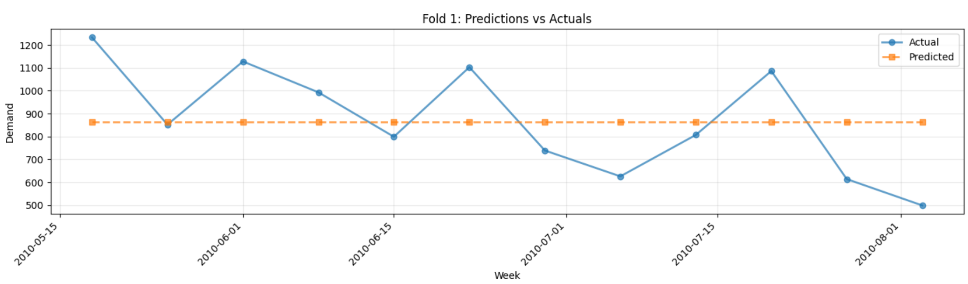

Figure 11: CV Time-Series (Fold 1)

One validation fold shows markedly larger gaps consistent with a regime shift (e.g. promotion/holiday) not fully encoded in features whereas the others are tighter with near-zero average bias, see figure below as well as appendix [37] and [40]. Overall, the fold plots indicate stable baseline fit with occasional under-prediction of spikes and mild smoothing of rapid changes.

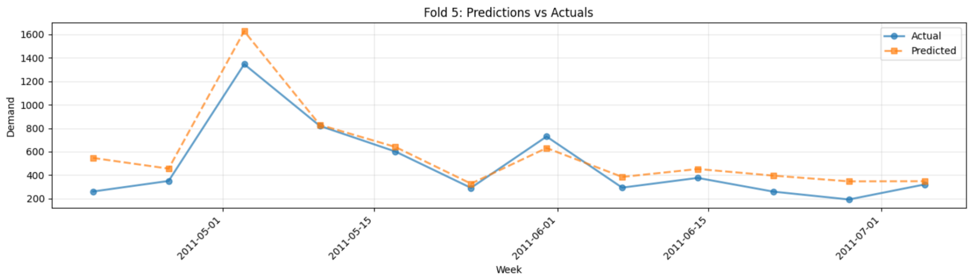

Figure 12: CV Time-Series (Fold 5)

Across folds the mean validation MAE ≈ 166 units and MAPE ≈ 30%. Dispersion across folds is moderate, indicating a reasonably stable learner, but one fold exhibits materially higher error consistent with a regime shift (e.g. promotion, calendar shock, or price change not fully learned). The residual mean is near zero (≈ +2 units), so there is no systematic over- or under-prediction on average.

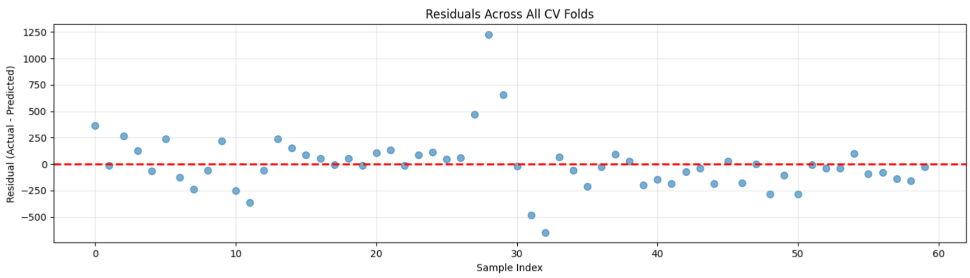

Figure 13: Residuals across all folds

The residual distribution above is asymmetric with several large positive spikes (actuals ≥ predicted) which signals that the current specification underestimates extreme peaks.
From a qualitative point of view the prediction generally follows level and short-term dynamics while missing some patterns during surges. Those misses are typical when peak drivers (promotional flags, sharp price moves, event dummies) are not explicitly encoded or when the model's objective/transformations down-weight tails. For the two-step sequential approach these findings have clear operational implications:
Near-zero bias is desirable here. There is no persistent drift that would accumulate into chronic overstock or stockouts. Under-prediction of peaks increases the risk of stockouts in surge weeks. In the current controller this risk is partly mitigated by (1) service-level-based underage cost in the inventory step, (2) warehouse safety-stock protection and (3) price control (corridor + profit criterion) that can temper demand when supply is tight.

Given that (1) the pipeline already encodes service-level constraints and safety stock, (2) the forecast is unbiased with moderate absolute error and (3) the dynamic pricing layer can partially ration demand, this baseline is fit-for-purpose as a first driver for the sequential and unified models.
Before transitioning to the downstream implementation of the demand forecast, it is essential to fit the XGBoost forecast method onto the whole dataset and forecast for the set time horizon. Fitting the final model on the whole dataset ensures that all information is used to estimate parameters, lowers variance and produces the model that is actually deployed.

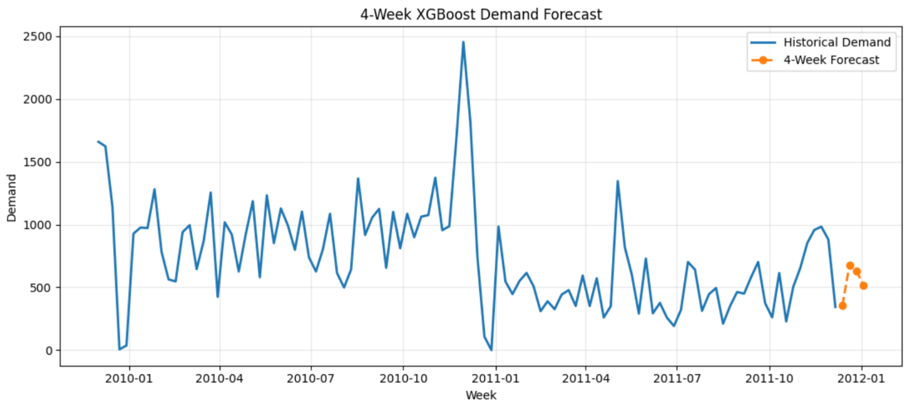

Figure 14: XGBoost forecast for 4 consecutive weeks

The short-horizon projection is mean-reverting and stays within the mid range of the recent history without reproducing the extreme spikes seen earlier. Weekly patterns seem smooth, which is consistent with the model's tendency to attenuate peaks observed in cross-validation as seen previously.
What this behavior might imply for later models: (1) safety stock and pricing corridors remain the main defensive methods against unexpected uplifts and (2) price variations will most likely be modest unless inventory is tight (shifting price due to policy rules). The trained model was saved for later use by exporting the forecaster as four artifacts which reproduce the exact prediction state:

- **xgb_model.json:** the full XGBoost booster (trees, scores, parameters).
- **feature_cols.json:** the exact feature list and order the model expects at inference. Your simulator must build features in this order.
- **meta.json:** metadata needed to rebuild features consistently: target name, price/log-price columns, demand lags, rolling windows, and the date column.
- **feature_builder.py:** leakage safe, single-row feature builder, given historical demand that reconstructs lags/rolling/seasonality to produce one-row vector expected by the model.

These files are the basis upon which the later sequential and unified models will be built on. The features such as week t are fixed in the artifacts ensuring full reproducibility.

8 Sequential: Two Step Heuristic Approach

8.1 Step 1: Setting up the Framework

This section explains the necessary steps taken to build the sequential two-step optimization model. The main objective is to combine a price-responsive demand forecast with operational rules (inventory) taking into account: capacity, safety stock and service levels targets so that the resulting decision operates in a realistic and comparable environment.
A week-by-week demand model is reconstructed based on the XGBoost forecast 7.5. The retail price p_t is defined in week t and $X_t(p_t)$, a feature vector that is filled with potential candidate price p_t into the current history. Based on those variables the full pipeline is constructed including: lags, rolling windows, seasonality, price lags. The pretrained baseline model produced the raw forecast which is progressively calibrated on a trailing window C to match the most recent scale, while preserving its shape:

$$D_t(p_t) = \max\{0,\ \text{CAL} \cdot f_\theta\big(X_t(p_t)\big)\}, \qquad \text{CAL} = \frac{\sum_{i \in C} y_i}{\sum_{i \in C} f_\theta(X_i)}.$$

This step is especially important because the model cannot be allowed to peek forward, only allowing it to look back into the historic context and then one step forward with the current forecasted demand window (rolling window). This rolling statistics to avoid peeking is expressed as:

$$\text{roll\_mean}_W(t) = \frac{1}{W} \sum_{k=1}^{W} D_{t-k}, \qquad \text{roll\_std}_W(t) = \sqrt{\frac{1}{W-1} \sum_{k=1}^{W} \big(D_{t-k} - \text{roll\_mean}_W(t)\big)^2}\,.$$

Additional business rules are added prior to the main code, ensuring realism and hindering the model to "explode" with optimal prices and order quantities. These business rules or Global settings include:

- **Service level target (SL):** The optimal operational performance that should be reached, ensuring quality of service: here defined at 90%.
- **Warehouse and Safety Stock (CAP):** The warehouse capacity is capped at 2.000 units, including a safety stock of 25% which aligns with conventional safety stock rules.
- **Lead Time (L):** Set identically to lead time in the synthetic inventory chapter: 2 weeks or L=2.
- **Guardrails:** A price corridor ensures that the optimization is kept within a certain threshold to avoid exploding prices.

8.2 Step 2: Operational Rules

In a second step, operational rules (inventory policies) for stock flow have been constructed to optimize supply with current demand and global settings. A sellable supply is defined with a protected safety stock using inventory information at the start of a week t as inv_t. The sellable supply is expressed as:

$$\text{sellable}_t(p) = \begin{cases} \min\big(\text{inv}_t,\ \text{cycle}\big), & \text{if } \widehat{D}_t(p)\ <\ \text{cycle}\ +\ \alpha\,\text{safety}, \\ \min\big(\text{inv}_t,\ \text{CAP}\big), & \text{otherwise.} \end{cases}$$

That rule ensures the safety stock buffer and only makes use of it when demand is exceptionally high and exceeds a threshold. Economic accounting is included as well, to compute the weekly profit $\pi_t(p)$ based on price p and weekly demand (current demand window). The weekly sales and profits are computed as:

$$s_t(p) = \min\big\{\text{sellable}_t(p),\ \widetilde{D}_t(p)\big\}.$$

$$\pi_t(p) = p\,s_t(p) - \text{COGS}(p)\,s_t(p) - C_o(\text{inv}_t - s_t(p)) - \beta_{\text{short,price}}\left[\widetilde{D}_t(p) - s_t(p)\right]_+.$$

A penalty system is also included to rectify the optimal order quantity and price steps:

- **Shortage Penalty:** An additional cost computed each forecasted week per missing unit, if inventory couldn't match the high demand.
- **Leftover Penalty:** An additional cost computed each forecasted week per leftover unit, if inventory was left with too many units.

These two metrics actively influence the decision of the model and penalize it, forcing it to drive the price down (in case of too many units left, because too much was ordered) or drive the price up (trying to curb demand).

8.3 Step 3: Price Optimization

For the next step, a pricing system is defined, respecting both demand response and operational constraints showed earlier. First, a price corridor is set up as guardrail to keep the optimization within the allowed thresholds. The upper and lower bounds of the corridor are defined as:

$$P_5 = \text{quantile}_{0.05}(\text{price}), \qquad P_{95} = \text{quantile}_{0.95}(\text{price}),$$
$$\text{PRICE\_MIN} = \max\big(10^{-6},\ 0.9\,P_5\big), \qquad \text{PRICE\_MAX} = 1.25\,P_{95}.$$

Next, a sell-through targeting rule is applied, seeking a candidate price, that will be able to clear the stock (up to the safety stock level):

$$D_t\big(p_t^*\big)\ \approx\ \text{sellable}_t\big(p_t^*\big)$$

However, if the none of the corridor edges are hit, the price falls back to a secondary pricing rule. In case demand is more or less flat, price decision is as follows:

$$p_t^* \in \arg\max_{p \in \mathcal{P}_t} \pi_t(p)$$

The crucial part of this section is to feed the chosen pair of optimal price $(p_t^*, \widehat{D}_t(p_t^*)$ back into the historic context so that all subsequent weeks inherit from that decision. This ensures that the models maintains coherence.

8.4 Outputs & Evaluation

Applying all those steps respectively outputs the following inventory-demand relationship The policy trajectory below highlights how the sequential policy handled the optimal order quantity and pricing problem.

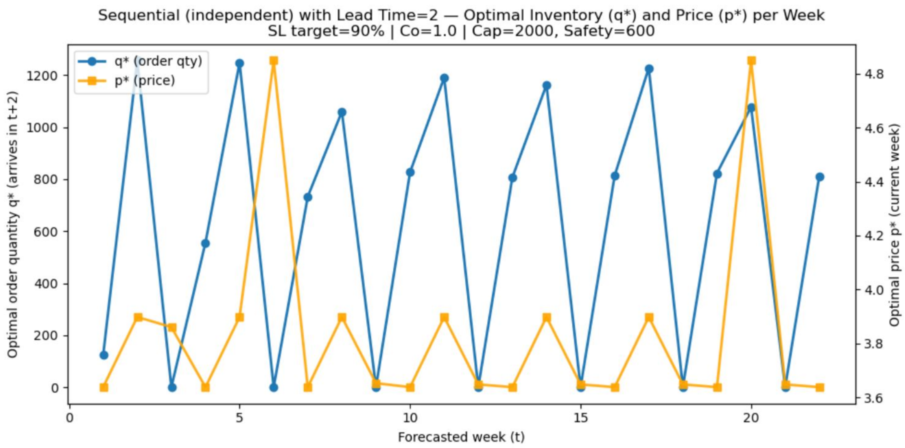

Figure 15: Policy Trajectory

The inventory order q_t^* displays a strong ordering delay pattern that is consistent with the set-up lead time of two weeks and the newsvendor target. The ordered quantities regularly spike between 1.200 and 1.250 units and then drop to zero. This behavior might indicate that the projected pipeline receipts plus on-hand inventory periodically suffice to cover expected demand. The optimal pricing p_t^* stays mostly in the corridor between £3.6-3.9 with some isolated peaks where the pricing abruptly tries to adjust to the demand.

Business outcomes over horizon

The main business KPIs for the sequential model are summarized in the table below.

KPI	Value
Total revenue	52.702,44
Total COGS	36.891,71
Total leftover penalty	2.505,00
Total shortage penalty	4.748,00
Total profit	**8.557,73**
Units sold	14.040,00
Fill-rate (proxy)	91.73%
Stockout rate (proxy)	40.91%
Inventory turns	18.67×
Avg weekly leftover units	113.86

Table 1: Business KPIs over planning horizon

The total profit is recorded at £8.557,73 at the end of the forecasting horizon with 14.040 units sold. The model also recorded losses due to penalties, specifically shortage penalties, amounting to £4.748 and an overall leftover penalty of £2.505.
From an operational standpoint, the fill-rate averages at 91.73% and a stockout rate (weeks recording unmet demand under sellable-supply) at 40.91%. An average weekly leftover of 113,86 units has also been recorded, highlighting some issues in planning steps.

Price behavior

The line graph below displays the price level behaviour of the sequential model. During two weeks the price hits the upper bound and never hits the lower bound. Most price ramps remain under the 10% threshold with some exceptions in weeks 6-7 and 20-21 as pointed out earlier.

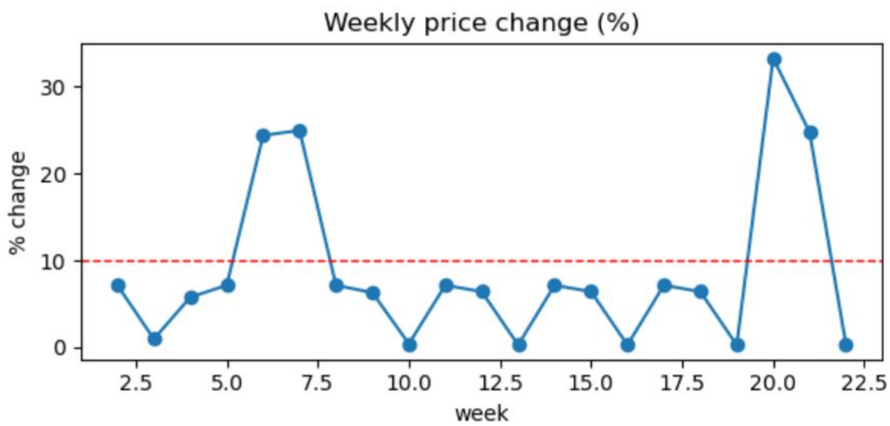

Figure 16: Sequential Model Price Level

Service Level
The Service Level (graph below), strongly oscillate between weeks, showing some difficulties in keepin consistency. Some weeks show very weak service level especially in the first weeks and an increasingly high (even hitting 100% SL) in the later weeks. Even though the behaviour is not consistent it averages to an average SL of 91.73% which is consistent with the fixed goals.

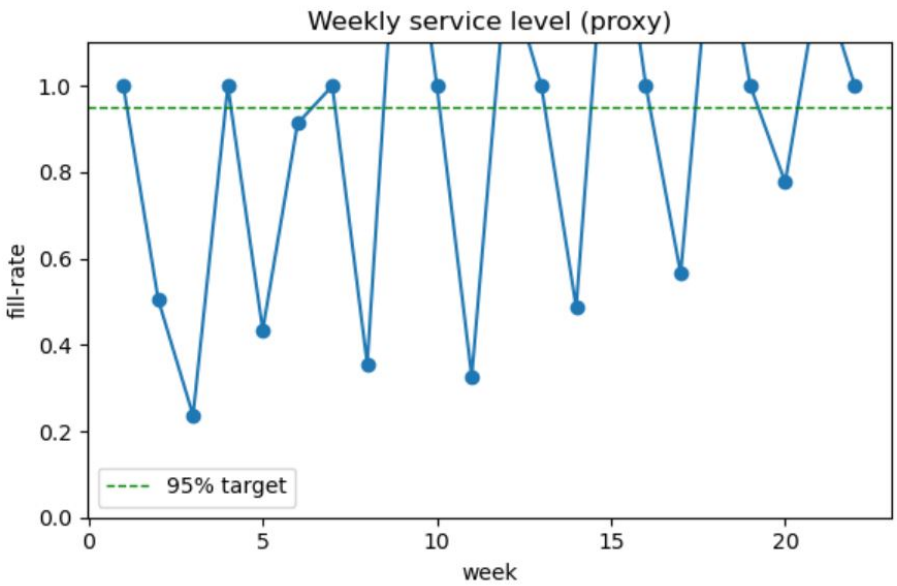

Figure 17: Sequential Model Service Level

This chapter created a clear outline on how a sequential controller can be set-up and operated. The simple step by step structure aims to be easily reproducible and output the necessary visualizations to verify the models performance along with the guardrails. The next chapter will apply the same pipeline and guardrails but selects (p_t,q_t) jointly within a single weekly program.

9 Joint Control Policy (Integrated Approach)

9.1 State Representation

This section shall serve as a quick recap showcasing which elements were recycled from the Two-Step Sequential Model and integrated into the Joint Control Policy. The rationale behind this structure was to create a reproducible and simple path for more transparency. The shared infrastructure consisted of the following elements:

- **Trailing Window Calibration (CAL) (see 8.1)**
- **Price Corridor Guardrail (see 8.3)**
- **Profit Accounting (see 8.2)**
- **Sell-Through Pricing (see 8.3)**
- **Rolling Window Stats (see 8.1)**
- **Sellable Weekly Supply (see 8.2)**

Having these elements already set-up from the previous step greatly improves building speed of the joint optimization. These steps also ensure a leakage safe, rolling feature and calendar encoding for the unified model. Like the sequential model it enables the operator to inject weekly price and price lags into the historic and current context. Similar to the necessary operations, the Global settings are also identical to those of the sequential approach (see chapter 8.1).

9.2 Sell-Through Price & Newsvendor

However, the joint optimization controller also approaches some aspects differently in terms of computing logic. Even though the model also uses sell-through price plans, it does so while anticipating at t+1, effectively projecting the inventory to the start of the next week as such:

$$\mathrm{inv}_{t+1}^{\mathrm{start}} = \min\bigl(\mathrm{CAP},\ \mathrm{inv}_t - s_t(p) + \mathrm{receipt}_{t+1}\bigr).$$

Within that projection, a receipt pipeline tracks the arrival schedule (L =2):

$$D_{t+1}\left(p_{t+1}^{\mathrm{plan}}\right) \approx \mathrm{sellable}_{t+1}\left(p_{t+1}^{\mathrm{plan}}\right).$$

This arrival schedule uses bisection inside of the price corridor (see chapter 8.3) setting brackets to the sellable inventory with regards to the anticipated inventory. That logic enforces the rule of low pricing if demand is too weak and high pricing if demand is too strong. This step is particularly important since it sets the basis for the Newsvendor engine. Based on the demand forecast, the Newsvendor chooses an order quantity that will arrive at $t + 2$. An inventory pre-order is denote (based on simulated sales and earlier receipts).

This pre and post ordering relationship gives basis to the per-unit overage cost Co, which is the holding penalty used in pricing. Its counterpart underage cost Cu is constructed to embed both margin and shortage penalty such as:

$$\text{margin}(p) = p - \text{COGS}(p), \tag{14}$$

$$C_u^{\text{target}} = C_o \cdot \frac{\text{SL}}{1 - \text{SL}}, \tag{15}$$

$$\beta_{\text{short,inv}} = \max\Big\{\beta_{\text{business}},\ C_u^{\text{target}} - \text{margin}(p)\Big\}, \tag{16}$$

$$C_u = \text{margin}\Big(p_{t+2}^{\text{plan}}\Big) + \beta_{\text{short,inv}}. \tag{17}$$

Putting all those elements together results in the newsvendor program:

$$q_t^* \in \arg\min_{q \geq 0}\ C_o \left[\,\text{inv}_{t+2}^{\text{pre}} + q - d_{t+2}\,\right]_+ \ +\ C_u \left[\,d_{t+2} - \left(\text{inv}_{t+2}^{\text{pre}} + q\right)\,\right]_+ \tag{18}$$

This program solves the problem of pre-post ordering while respecting capacity constraints and service level SL.

9.3 Joint Optimization Step

The joint controller uses a future over and under cost (at $t + 2$) which is used by the newsvendors order optimal at a certain planned price:

$$\text{FOC}_{t+2}(p) = C_o \Big[\,\text{inv}_{t+2}^{\text{pre}}(p) + q_t^*(p) - d_{t+2}(p)\,\Big]_+ \ +\ C_u \Big[\,d_{t+2}(p) - \left(\text{inv}_{t+2}^{\text{pre}}(p) + q_t^*(p)\right)\,\Big]_+, \tag{19}$$

$$\text{where} \quad [x]_+ := \max\{x, 0\}. \tag{20}$$

This step selects a potential "attractive" looking price for a present week. But it would also enforce an expensive mismatch if the specific order if penalized via $\lambda FOC_{t+2}(p)$. In other words, this term combines today's price to the future inventory features (lead time, capacity, safety stock), preventing too shortsighted discounting (even if myopic approach). It would also prevent overpricing that would create high overstock costs or stockouts two weeks later ($t + 2$) that would drive up leftover or stockout penalties.
To address issues of potential abrupt jumps, despite price corridor and FOC, the controller implements a smoothness step such as:

$$\text{SMOOTH}_t(p) = p(p - p_{t-1})^2, \tag{21}$$

$$p \geq 0. \tag{22}$$

Combined, the "joint" score of those terms is calculated as:

$$\mathrm{J}_t(p) = \pi_t(p) - \lambda \mathrm{FOC}_{t+2}(p) - \mathrm{SMOOTH}_t(p). \tag{23}$$

The profit term (see chapter 8.2) rewards immediate profits keeping in mind the safety sellable rules. FOC penalizes "todays" price if it would force expensive over/under at the time the order arrives and the smoothness step evens out the pricing to avoid unrealistic price volatility that might result from previous mistakes. All those steps combined with the inventory ordering and capacity rules, as well as pricing rules should ensure a tighter coupling between price and order rather than perceiving them separately.

9.4 Outputs & Evaluation

Putting all those elements together, the model outputs the following results for the forecast horizon. How the policy trajectory behaves can be seen in the graph below:

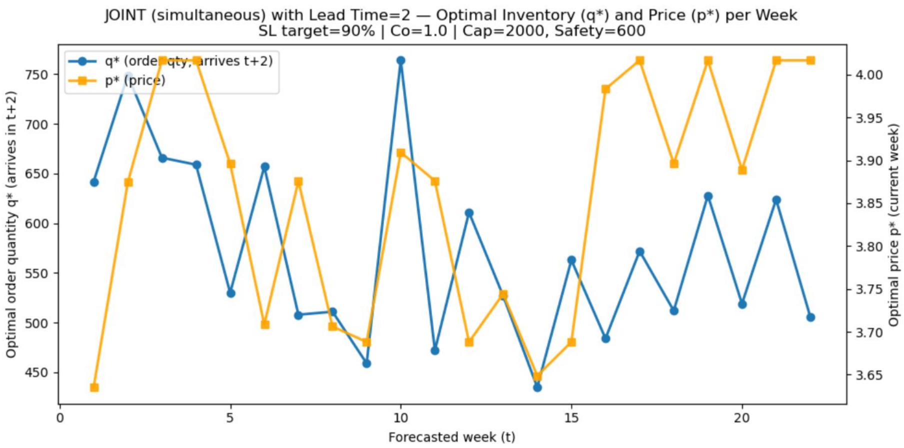

Figure 18: Joint Model Performance

The weekly optimal order quantity q_t^* is moderate and stable with roughly 450-750 units. Even though one spike can be seen around week 10, q_t^* manages to adjust smoothly around a mid-level order quantity. This shape is the product of the joint policy, choosing price and inventory together, and setting penalties for projected under/over at delivery. This rule dampens the typical oscillations of orders.

The optimal pricing mechanics p_t^* shows the coordinated price path, reacting to changes int q_t^*. As can be seen in the graph, whenever the chose price implies a higher expected demand, the optimal order quantity rises and when the price implies demand braking, optimal q_t^* falls. This produces the anti-cyclical pattern that can be seen on the graph.

Business outcomes over horizon

The financial outcome over the horizon can be summarized in the table below:

KPI	Value
Total revenue	49.240,75
Total COGS	34.468,52
Total leftover penalty	525,00
Total shortage penalty	1.138,00
Total profit	**13.109,22**
Units sold	12.810,00
Fill-rate (proxy)	95.75%
Stockout rate (proxy)	40.91%
Inventory turns	21.13×
Avg weekly leftover units	23.86

Table 2: Business KPIs over planning horizon: Joint Model

Over the planning horizon, the joint policy generated a total revenue of 49.240,75£ against 34.467,52£ in COGS. The total leftover penalty for the horizon amounts to £525 and the shortage penalty to £1.138, overall £1.663. The aggregate profit for that period adds up to £13.109,22 with 12.810 units sold. Additionally the inventory has been completely cleared out at the end of the period.

Price behavior

In terms of pricing behavior, the figure below displays the optimal pricing evolution inside the corridor over the horizon:

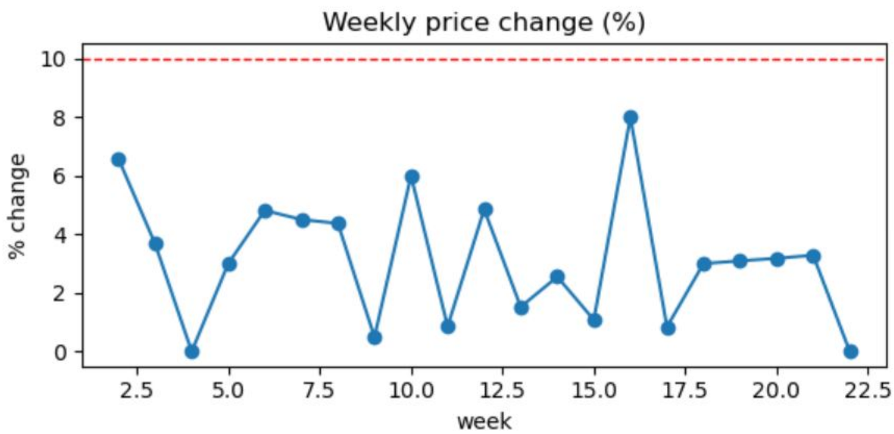

Figure 19: Joint model price level

The optimal prices p_t^* steadily remain within the corridor over the whole period. The indicator of the price hitting the upper bound is zero in every week and the weekly price changes remains well below the 10% ramp limit. Toward the late horizon the price experience a mild upward drift while order quantities remain moderate. This is however consistent with the higher forecasted demand elasticity at lower prices as well as the security step in protecting the inventory orders while making sure the safety stock remains untapped.

Service Level

In terms of service level performance, the fill-rate averages at roughly 96% with some small dips in the early and mid-horizon. The stockout rate is at 41%, indicating frequent but typically small shortfalls.

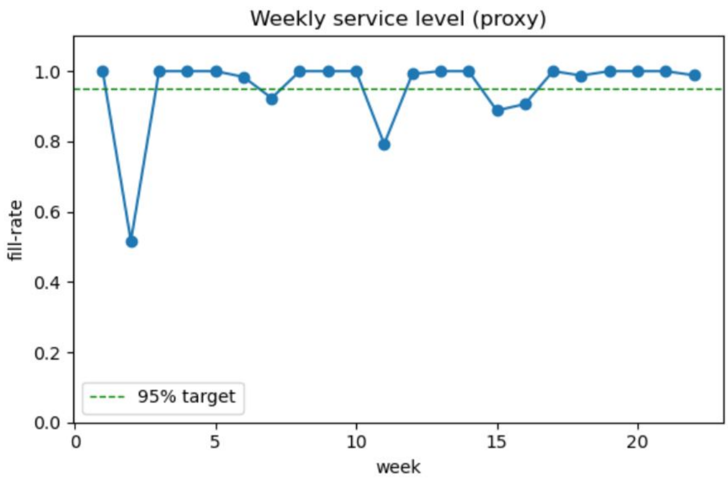

Figure 20: Joint model service level

Overall this graph shows consistent low shortage penalty total and high overall fill-rate. To summarize the joint optimization model is able to deliver coherent, operationally feasible pricing inventory planning over the test horizon. The price remains within the corridor and below the 10% limit, the policy achieves the targeted service level as well as low average leftover units. The coupling of price and newsvendor based ordering works smoothly and predictably.

Although the model displays some hints of monotonicity violations, the evaluation shows that the joint policy approach aligns with the set rules. The next chapter will focus on directly comparing the outputs of the two models and drawing first conclusions about the differing performances.

10 Model comparison

This section aims to paint the contrast between the sequential two-step policy against the joint policy. First a technical comparison will be performed by dissecting how the algorithms work under identical input to highlight where the code paths diverge. Afterwards, the model performances will be compared using the common business KPIs that were outputted by the models in earlier chapters:

10.1 Technical Comparison

This section serves as a ground for technical comparison between the two models. The chart below provides an overview of how the two models act and process the information flows in 6 steps:

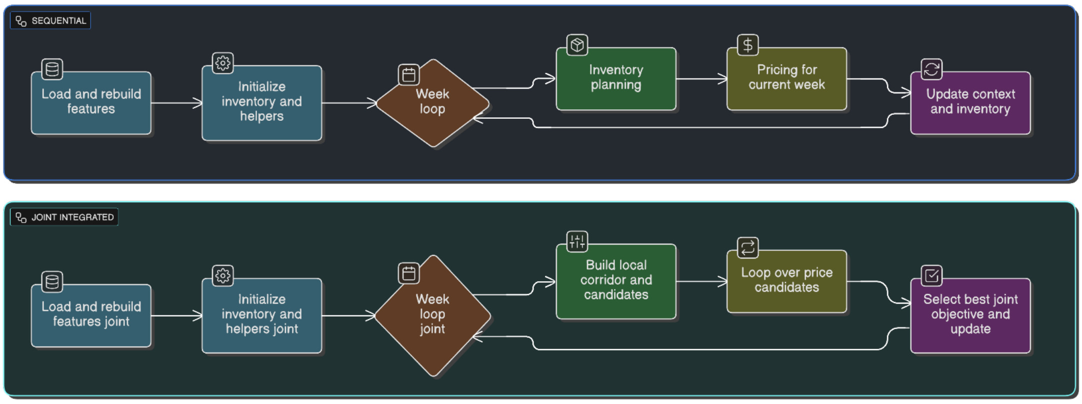

Figure 21: Sequential vs Joint Process Flow

As depicted above, the model structures are identical at first until the fourth step or after the weekly looping. The key differences are the decision timing, price candidate set, inventory cost coupling, future costs and context evaluation for demand features.

Decision Timing

In the 4th to the 5th step, the sequential model first chooses the order q_t, using a newsvendor step at a certain planning price. Only after that q_t is fixed, the model chooses the sellting price p_t in order to maximize the current week profit with regards to capacity and saftey rules. The two choices are architecturaly and temporally decoupled.

The join model on the other hand starts by simulating the next two weeks for each candidate price p_t, computing the optimal newsvendor order for $t + 2$ and with that price path, evaluating a single objective. That objective is trading off the current profit and future mismatch penalties. The optimal price p_t^* and order quantity q_t^* are then selected together.

Price Candidate Set

In the 5th step of the sequential model, the prices are taken from a wide historic-driven context grid based on the global corridor [PRICE_MIN,PRICE_MAX]. Whenever possible, a sell-through bisection selects a price, that approximately clears the cycle stock. Otherwise a price is picked that maximizes profit at that point.

The joint model approaches the candidate price differently in a much more localized and adaptive manner. A corridor it centered around the last executed price p_{prev} (e.g. +/- 10%). Then a sell-through seed is computed within that band and a refined grid around that seed is built. A quadratic term $\mu(p - p_{prev})^2$ smooths the week to week movements, keeping the search centered to avoid extreme decisions.

Inventory Cost Coupling

In the Sequential model the service level target is decoupled from current week inventory status such as shortage penalty. The SL only enters via $C_u^{\text{target}} = C_o \cdot \frac{1-\text{SL}}{\text{SL}}$. There is no relinking between price and SL. The joint model anticipates a sell through price plan for $t+1$ and $t+2$ for every candidate price p_t. The implied margin at t+2 is computed and sets the newsvendor underage cost $C_u = margin(p_{t+2}^{\text{plan}}) + \beta_{short,inv}$. This way the order q_t is defined by the cadidate price path coupling inventory and pricing together.

Future Cost Term

The future cost term is one of the unique and key differences that is only part of the architecture flow of the joint model (Step 5-6). As pointed out in earlier chapters, the joint policy includes a penalty on the expected over/under at future $t+2$ by choosing price p_t and the resulting order $q_t^*(p_t)$:

$$\mathrm{J}(p_t) = \pi_t(p_t) - \lambda\,\mathrm{FOC}_{t+2}(p_t) - \mu\,(p_t - p_{\text{prev}})^2. \tag{24}$$

The λ term is the main knob to control the hedging for future mismatch risk or favoring immediate margin for lower λ values.

Context Evolution

Both approaches send back the triplet {week t, $\widehat{D}_t(p_t^*)$, p_t^*} so that the next weeks lagged feature as well as rolling windows and price_lag_1 used for the first weeks are correct. The only nuanced difference is as stated earlier. The hypothetical anticipated $(p_t, \widehat{D}_t(p_t))$ to keep consistency over $t+1$ and $t+2$. It is only after picking the best possible candidate in the last step that the final triplet is appended back.

10.2 Comparing Results

Following the technical comparison it is now essential to investigate the impacts that the technical differences had on the actual Business results for the two models. The table below summarizes the main Business KPIs of the two models directly including a delta column to highlight the differing results:

Metric	**Sequential**	**Joint (integrated)**	**Δ (Joint−Seq)**
Revenue	52.702,44	49.240,75	−3.461, 69
COGS	36.891,71	34.468,52	−2.423, 19
Leftover penalty	2.505,00	525,00	**−1.980, 00**
Shortage penalty	4.748,00	1.138,00	**−3.610, 00**
Profit	**8.557,73**	**13.109,22**	**+4.551,49**
Units sold	14.040	12.810	−1.230
Ending inventory	210	0	−210

Table 3: Business outcomes over the planning horizon.

Clearly, the joint policy generates substantially higher profits despite overall lower revenues. This can be traced back to step of cutting mismatching costs more aggressively such as: shortage penalties drop by roughly 76% and leftover penalties by 79%. Even though fewer units have been seld in the Joint approach, they are sold with better alignment with supply and capacity. Stockouts and leftover that decrease margin are avoided.
Additionally, in terms of efficiency, some clear differences can be seen such as the fill rate 91.73% (sequential) vs. 95.75% (joint). Clearly the joint model meets the service target much more consistently across all weeks. Even though the sequential model reaches 100% during some weeks, the key patterns here is consistency.
The Stockout rate is identical in both models around 41%, however the magnitude of shortfalls is smaller in the joint policy, reflecting the large drop in shortage penalties. This drop is particularly important since this context considers that it more favourable to have a lower shortage penalty (still inventory available) rather than high leftover penalty which could eventually lead to customers churning. With a difference of £3.610 of shortage penalty, the joint model strongly mitigates the churn risk.
The inventory turnover also shows some clear improvement from 18.76x vs 21.13x, indicating that more inventory gets moved week to week and less ending inventory is left. The model capacities are used more efifciently in the joint model compared to the sequential model.
The price path diagnostics also show clear differences between the sequential model, hitting the upper corridor twice (in week 6 and 20) and the joint model that actually never hits the upper bound. The operational superiority of the joint model can also be seen in the weekly price ramp (see chapter 8.4).

While the sequential price shows spikes exceeding the 10% limit, the joint policy stays below 8% which is a direct positive effect display of the local corridor and smoothness penalty. To summarize all the points above, the joint model clearly displays that an integrated price and inventory decision approach can potentially yield more profit-efficient operations under exactly the same business rules.
These methods proved Federgruen and Heching (1999) approach right in using dynamic pricing and bi-directional price change for optimal profits. Also the works of Hasiloglu-Ciftciler and Kaya (2025) and Kumar et al. (2024) proved these approaches effective for business contexts especially the necessity of incorporating inventory policies (Kumar et al., 2024) . It is important to point out that these results rest on very restricted and monitored sets of rules and policies (such as price corridors). The next chapter shall discuss and examine the limitations and robustness considerations of those models.

11 Limitations & Discussion

This section focuses on clarifying the limitations of the methods used in this thesis. It will also show how sensitive each model is to: elasticity misspecification, corridor design, penalty tuning and the use of synthetic inventory states and outlining practical implications for deployment at scale.

11.1 Data Quality and Representativeness

The main limitation of this thesis included the absence of inventory log reports. This lack of information initiated the creation of the synthetic dataset based on historic demand patterns. This however could only be done only in a hypothetical setting such as this thesis. Even though the synthetic inventory recreated seemingly realistic invetory information, the real inventory history of the online store could never be replicated in its wholeness. This has to be taken into consideration when analyzing the result. Real results might differ from this simulation.

The results are also heavily dependent on the set of strict business policies (lead time, penalties, reorder points etc.). Tweaking each of these global settings considerably changes the results and might differ based on the context. For example, a retailer selling perishable goods will most likely set very high leftover penalties since the products cannot be sold after a certain amount of time. The findings of other scholars also support the unified approach such as the work of Hasiloglu-Ciftciler and Kaya (2025) who also observed positive profits but for perishable goods.

The decision to focus on one SKU was also deliberate to keep model complexity low which lead however to very few observations to analyze and train the model with. This flaw could be straighten out in future work with less time constraints and more computational resources. A joint model with multiple SKUs could investigate cross-item purchasing patterns that could reveal additional information and improvements for inventory planning and pricing.

In terms of representativeness, even though the model uses only one SKU, it is constructed in a very modular way. To test the model on other products, the user only needs to input the corresponding product identifyier and define the key variables (if using another dataset) the model will still operate and output the results. Any other SKU could be selected from the entire UCI II Dataset and the model would perform as intended.

However, it might encounter some restrictions, not from a technical view, but from a statistical point of view since some products have very little observation. So if selecting another SKU to investigate potential profits, the user must question the validity of the result and if they are statistically relevant.

11.2 Elasticity Identifiability & Monotonicity

One noticeable issue was that both models repeatedly exhibited local non-monotonicity of the learned demand curve $D_t(p)$ in seemingly every week. The outcome is consistent with the used data and context. Since this study is looking at one single SKU, a limited historical price variation, strong interactions and lagged demand, these elements provide an insufficient experimental variation to identify a strictly negative own-price effect. If the base model (XGBoost) cannot catch those specific relationships, they get preserved in the local artifacts that will carry on this flaw into the controllers.

The monotonicity diagnostics further indicate that price elasticity is weakly identified and that price responses in the models should be interpreted carefully. The implemented guardrails stabilize decision making especially under rapidly changing demand. The optimizer is very sensitive to noise and can become unstable, as can be seen in the policy trajectories [15], [18] (see appendix) as well as the service levels [17] (see appendix).

The monotonicity issue is an inherent limitaiton of flexible predictive learners under limited price variations. This issue could be addressed by strengthening the model with monotonic constraints on price or hybrid parametric-tree demand. But under the time constraints and limtiations of this thesis, the present results still hold validity and should rather be read as policy performance under realistic data constraints rather than as precise elasticity measurement.

11.3 Scalability and Computational Constraints

This section points out computational differences and that the joint approach took considerably longer to compute ($\geq$ 8min.) against the sequential model ($\leq$ 1min.).

Model	Block	Wall time (s)	Peak memory (MB)
Sequential	sequential_inventory_pricing	114.40	76.80
Sequential	fixed_diagnostics	53.36	0.96
Sequential	policy_validation	73.93	4.78
Joint	joint_inventory_pricing	960.84	76.99
Joint	validation_step	69.09	4.88

Table 4: Wall-clock time and peak memory usage by major notebook blocks

All experiments and simulations were executed with the following components, see appendix [49]. This configuration is modest and representative of a constrained non-GPU setting. The table 4 (above) summarizes the performance (runtime and memory) of both models under the system constraints. They use up the same amount of memory to run the code.

In terms of processing time, the joint model takes more than 8 times the processing time, indicating considerable time limitations. This behaviour was expected since each week evaluates a set of candidate prices and for each candidate simulates inventory propagation,

a sell through price $t + 2$ and solves a newsvendor order, all within a weekly loop.
The computational ceiling has strongly shaped the scope of this study. The results should be viewed as lightweight and operationally deployable controllers under realistic compute. The sensitivity to sudden demand shifts and diagnostic monotonicity reflect compute-driven approximations.
Scalability to the full set of available SKUs would be possible for future research but it would require both algorithmic and system upgrades. Potential updates might include:

Algorithmic:

- Batch demand evaluation for all candidate prices
- Gradient based price search using local elasticity estimes
- Multi SKU decomposition with Lagrangian coupling for shared capacity

Systems:

- GPU acceleration for XGBoost
- Multicore parallelization across SKUs and weeks
- Larger RAM capacities to carry out multi-SKU analyses

Alltogether, it can be said that flexible learners such as XGBoost may exhibit local non-monotonicities that can be stabilized by introducing guardrails. Furthermore the results are shaped by explicit managerial business oriented choices: service level targets, shortage and leftover penalites, lead time and capacity.
These parameters are very context dependent and influence the ordering engine heavily. Computational constraints reinforce these trade-offs, especially the joint controller being substantially more expensive to run than the sequential baseline framework. Nonetheless both were build in a transparent, lightweight and reproducible manner for use on simple hardware.

12 Conclusion

Overall, this thesis can be understood as an attempt to redesign retail decision making in a realistic and interpretable way: providing a clear narrative and context, starting from a simple demand forecast pipeline up to guardrailed ML controllers that are able to translate demand forecasts into executable order-price choices under real operational constraints. The sequential controller and joint policies highlighted two crucial governance paradigms.

On one hand, the sequential controller is simple and is aligned with historic and current practices but at the cost of stock imbalance weak price stability and suboptimal service levels.

On the other hand, the joint controller treats price as a parameter for tweaking future inventory risk: maximizing the current profit and penalizing whenever deliver at the time is over or underserving. Both controllers acted under an identical set of business rules including: lead time, capacity constraints, safety stock and both controllers managed to generate feasible trajectories. The joint policy however displayed stronger economic performance and smoother price paths.

From a managerial perspective, the integrated model is not simply better but manages to show how value accures when optimization objectives mirror how costs actually materialize in real life operations. Leftover and shortage penalties become strong levers that hold revenue against resilience.

In an attempt to keep a realistic approach this study also implemented other operational parameters that real life retailers would expect: rolling calibration, corridor based guardrails to prevent extreme behavior, diagnostics to monitor service levels, price ramps and imperfect elasticity monotonicity. This study offers a deployable blueprint for retailers seeking decision-making support that is both data driven and policy aware.

The limitations of this work are clear: single SKU, synthetic inventory, finite price variation all mean that elasticities are poorly identified and the results should be read as more of a policy performance under realistic constraints rather than a highly precise prediction tool. Future work could use this study as a baseline to further develop the joint model by extending it to multi-SKU portfolio, promotional encoding, embedding monotonicity control parameters including competitor pricing behaviors or even collaborating with a retail partner to secure longer, richer histories (with true inventory logs) and run price experiments.

Within the research field of dynamic pricing literature, this study contributes not only as an operational bridge between flexible ML forecasts to actionable controllers but also as a clarification on how to align pricing with inventory risk when business stakes include not only margin, but service stability. So next time you are wondering why you received a different online shopping price than the previous day, now you know.

References

M. Aoki. *On Some Price Adjustment Schemes*, volume Volume 3 Nr.1, pages 95–115. NBER, sanford v. berg edition, 1974. URL `https://www.nber.org/system/files/chapters/c9998/c9998.pdf`.

D. L. Banks and S. E. Fienberg. *Data Mining, Statistics*, pages 247–261. Elsevier, 3rd edition edition, 2003. doi: 10.1016/B0-12-227410-5/00164-2.

G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5:203–229, 7 2003. ISSN 1523-4614. doi: 10.1287/msom.5.3.203.16031.

L. Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16, 8 2001. ISSN 0883-4237. doi: 10.1214/ss/1009213726.

K. C. Chan, M. Rabaev, and H. Pratama. Generation of synthetic manufacturing datasets for machine learning using discrete-event simulation. *Production & Manufacturing Research*, 10:337–353, 12 2022. ISSN 2169-3277. doi: 10.1080/21693277.2022.2086642.

M. Chen and Z. Chen. Recent developments in dynamic pricing research: Multiple products, competition, and limited demand information. *Production and Operations Management*, 24:704–731, 5 2015. ISSN 1059-1478. doi: 10.1111/poms.12295. URL `https://journals.sagepub.com/doi/abs/10.1111/poms.12295`.

J. Chu, P. Chintagunta, and J. Cebollada. Research note—a comparison of within-household price sensitivity across online and offline channels. *Marketing Science*, 27:283–299, 3 2008. ISSN 0732-2399. doi: 10.1287/mksc.1070.0288. URL `https://www.researchgate.net/publication/220659132_Research_Note-A_Comparison_of_Within-Household_Price_Sensitivity_Across_Online_and_Offline_Channels`.

A. de Palma, R. Lindsey, and S. Proost. Research challenges in modelling urban road pricing: An overview. *Transport Policy*, 13:97–105, 3 2006. ISSN 0967070X. doi: 10.1016/j.tranpol.2005.11.006.

A. V. den Boer. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20:1–18, 6 2015. ISSN 18767354. doi: 10.1016/j.sorms.2015.03.001.

F. Fang, T.-D. Nguyen, and C. S. Currie. Joint pricing and inventory decisions for substitutable and perishable products under demand uncertainty. *European Journal of Operational Research*, 293:594–602, 9 2021. ISSN 03772217. doi: 10.1016/j.ejor.2020.08.002. URL `https://eprints.soton.ac.uk/430624/1/Joint_Pricing_and_Inventory_Decisions_for_Substitutable_and_Perishable_Products_under_Demand_Uncertainty_EJOR.pdf?`

A. Federgruen and A. Heching. Combined pricing and inventory control under uncertainty. *Operations Research*, 47:454–475, 6 1999. ISSN 0030-364X. doi: 10.1287/opre.47.3.454.

G. Gallego and G. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40:999–1020, 8 1994. ISSN 0025-1909. doi: 10.1287/mnsc.40.8.999.

T. E. Goltsos, A. A. Syntetos, C. H. Glock, and G. Ioannou. Inventory – forecasting: Mind the gap. *European Journal of Operational Research*, 299:397–419, 6 2022. ISSN 03772217. doi: 10.1016/j.ejor.2021.07.040. URL `https://www.sciencedirect.com/science/article/pii/S0377221721006500`.

M. Gupta. Dynamic pricing optimization in e-commerce using ai and machine learning: A comprehensive framework for demand prediction and revenue maximization. 2 2025.

M. Gupta and J. Jain. Optimizing e-commerce dynamic pricing using aggregated market data and cloud-based analytics. *International Journal of Global Innovations and Solutions (IJGIS)*, 12 2024. doi: 10.21428/e90189c8.ed197a70.

M. Hasiloglu-Ciftciler and O. Kaya. Dynamic inventory control and pricing strategies for perishable products considering both profit and waste. *Computers & Operations Research*, 181:107103, 9 2025. ISSN 03050548. doi: 10.1016/j.cor.2025.107103.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55, 2 1970. ISSN 00401706. doi: 10.2307/1267351.

J. Huber and H. Stuckenschmidt. Daily retail demand forecasting using machine learning with emphasis on calendric special days. *International Journal of Forecasting*, 36: 1420–1438, 10 2020. ISSN 01692070. doi: 10.1016/j.ijforecast.2020.02.005.

IFRS. International accounting standards 2: Inventories. Technical report, International Accounting Standards Board, 2016. URL `https://www.ifrs.org/content/dam/ifrs/publications/pdf-standards/english/2022/issued/part-a/ias-2-inventories.pdf?bypass=on&utm_source=chatgpt.com`.

G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An introduction to Statistical Learning*, pages 1–13. Springer, 7 2023. doi: 10.1007/978-3-031-38747-0_1.

H. Kagalwala, G. V. Radhakrishnan, I. A. Mohammed, R. R. Kothinti, and N. Kulkarni. Advances in consumer research predictive analytics in supply chain management: The role of ai and machine learning in demand forecasting. Technical report, 2 2025. URL `https://acr-journal.com/`.

W. Kincaid and D. Darling. An inventory pricing problem. *Journal of Mathematical Analysis and Applications*, 7:183–208, 10 1963. ISSN 0022247X. doi:

10.1016/0022-247X(63)90047-7. URL https://pdf.sciencedirectassets.com/272578/1-s2.0-S0022247X00X03320/1-s2.0-0022247X63900477/main.pdf?

L. Kumar, K. Sharma, and U. Khedlekar. Dynamic pricing strategies for efficient inventory management with auto-correlative stochastic demand forecasting using exponential smoothing method. *Results in Control and Optimization*, 15:100432, 6 2024. ISSN 26667207. doi: 10.1016/j.rico.2024.100432.

S. Lindlahr, S. Zavialova, C. Duggan, and R. L. Re. Industries & markets ecommerce: market data & analysis. Technical report, Statista, 12 2024. URL https://www.statista.com/study/42335/ecommerce-report/.

J. Liu, Z. Pang, and L. Qi. Dynamic pricing and inventory management with demand learning: A bayesian approach. *Computers & Operations Research*, 124:105078, 12 2020. ISSN 03050548. doi: 10.1016/j.cor.2020.105078.

Y. Long, S. Kroeger, M. F. Zaeh, and A. Brintrup. Leveraging synthetic data to tackle machine learning challenges in supply chains: challenges, methods, applications, and research opportunities. *International Journal of Production Research*, pages 1–22, 1 2025. ISSN 0020-7543. doi: 10.1080/00207543.2024.2447927.

Y. Lu, L. Chen, Y. Zhang, M. Shen, H. Wang, X. Wang, C. van Rechem, T. Fu, and W. Wei. Machine learning for synthetic data generation: A review. 4 2025.

Q. Ma, S. Feng, and J. Liu. Dynamic pricing and demand forecasting: Integrating time-series analysis, regression models, machine learning, and competitive analysis. *Applied and Computational Engineering*, 93:149–154, 11 2024. ISSN 2755-2721. doi: 10.54254/2755-2721/93/20240935. URL https://www.researchgate.net/publication/385709574_Dynamic_pricing_and_demand_forecasting_Integrating_time-series_analysis_regression_models_machine_learning_and_competitive_analysis.

M. Nowak and M. Paw lowska-Nowak. Dynamic pricing method in the e-commerce industry using machine learning. *Applied Sciences*, 14:11668, 12 2024. ISSN 2076-3417. doi: 10.3390/app142411668.

K. R., S. Syed, P. M., A. E., N. S., and A. K. Dynamic pricing strategies using machine learning in e-commerce. *Advances in Consumer Research*, 2:1951–1960, 2025. ISSN 0098-9258. URL file:///C:/Users/cbram/Downloads/Dynamic_Pricing_Strategies_Using_Machine_Learning_in_E-Commerce_1.pdf.

I. Segal. Optimal pricing mechanisms with unknown demand. *American Economic Review*, 93:509–529, 5 2003. ISSN 0002-8282. doi: 10.1257/000282803322156963.

H. Sun. A literature review of e-commerce supply chain management. *BCP Business & Management*, 20:486–496, 6 2022. ISSN 2692-6156. doi: 10.54691/bcpbm.v20i.1023.

J. Thomas. Price-production decisions with deterministic demand. *Management Science*, 16:747–750, 7 1970. ISSN 0025-1909. doi: 10.1287/mnsc.16.11.747.

M. Wang, Y. Liu, G. Li, T. R. Payne, Y. Yue, and K. L. Man. Unlocking your sales insights: Advanced xgboost forecasting models for amazon products. 11 2024.

A. S. Whittemore and S. C. Saunders. Optimal inventory under stochastic demand with two supply options. *SIAM Journal on Applied Mathematics*, 32:293–305, 3 1977. ISSN 0036-1399. doi: 10.1137/0132023.

Z. Zhang, Y. Jiang, Q. Li, and A. Han. Otpto: Joint product selection and inventory optimization in fresh e-commerce front-end warehouses. 5 2025.

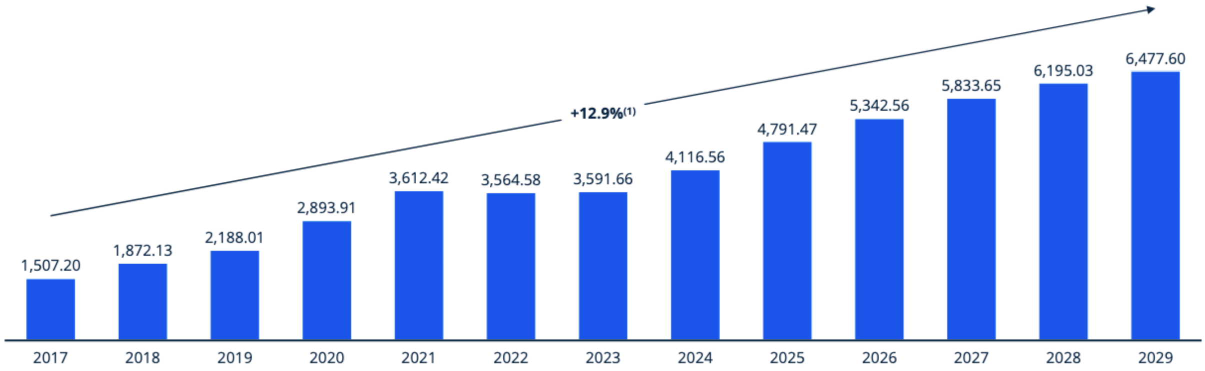

Figure 22: E-Commerce revenues are estimated to increase at a CAGR of 12.9% from 2017 to 2029, Source: Statista Market Insights 2024

Name	Type	Description
InvoiceNo	**STR**	Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.
StockCode	**STR**	Product (item) code. Nominal. A 5-digit integral number uniquely assigned to each distinct product.
Quantity	**INT**	The quantities of each product (item) per transaction. Numeric.
InvoiceDate	**DATETIME**	Invoice date and time. Numeric. The day and time when a transaction was generated.
UnitPrice	**FLOAT**	Unit price. Numeric. Product price per unit in sterling (£).
CustomerID	**STR**	Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.
Country	**STR**	Country name. Nominal. The name of the country where a customer resides.
Description	**STR**	Product (item) name. Nominal.

Figure 23: Online Retail II UCI Dataset (D. Chen, 2012) Non-altered

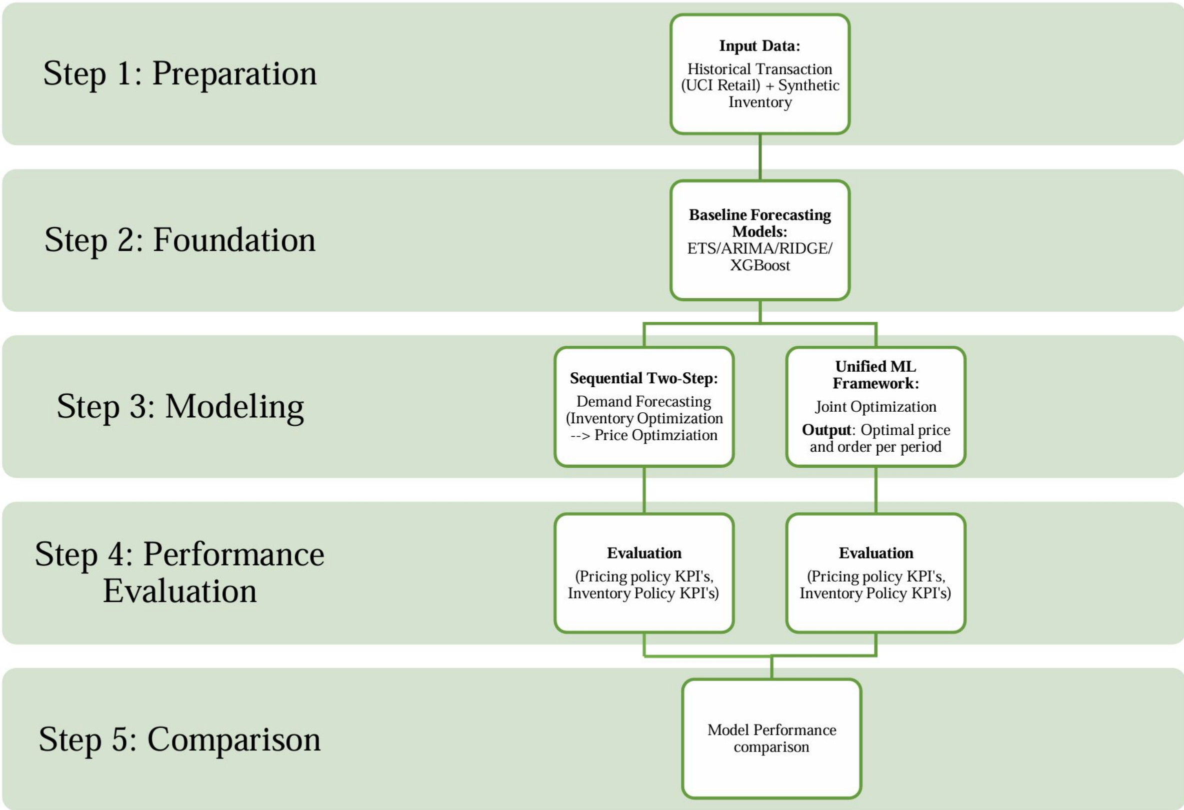

Figure 24: Schematic Overview: Full Methodological Framework

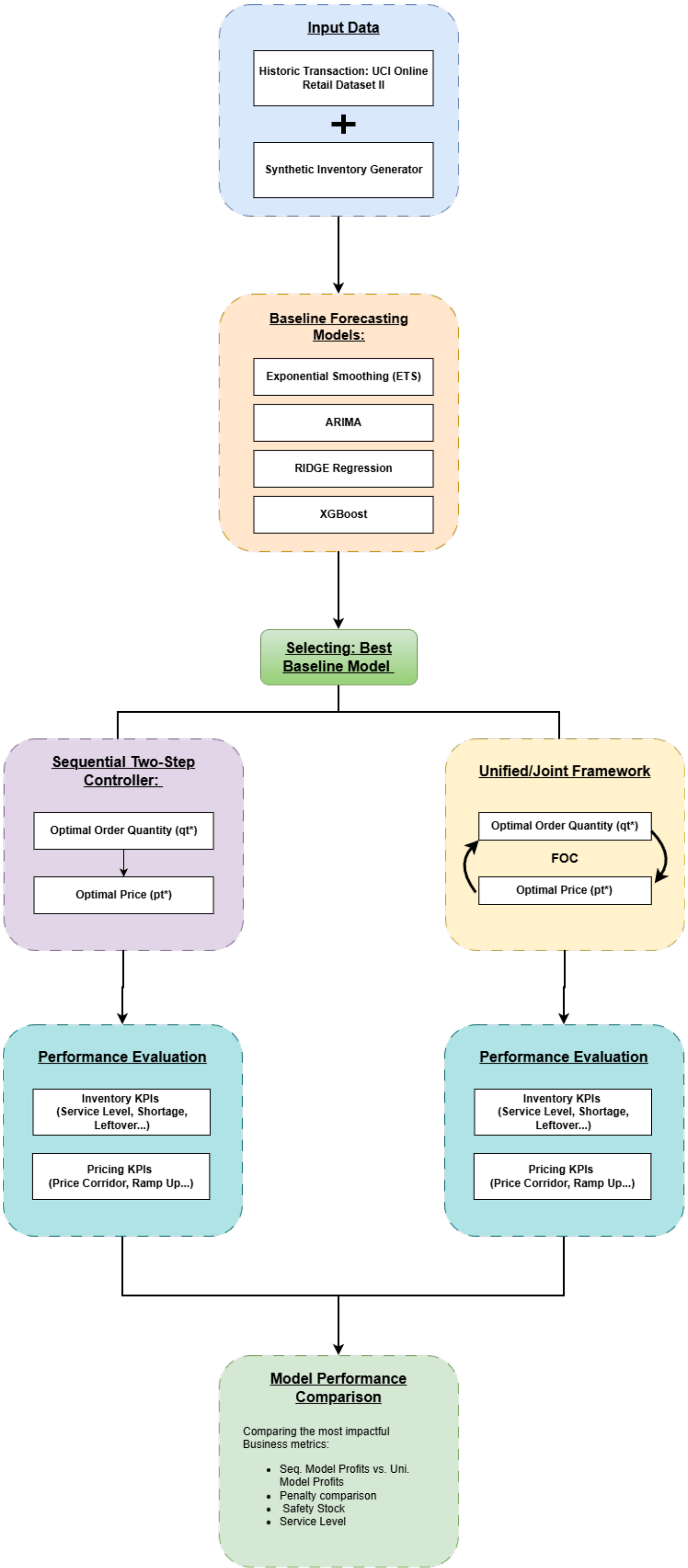

Figure 25: Schematic Overview: In depth Methodological Structure

Panel Preview — Synthetic SKU–Week Dataset (first 12 weeks)

WeekStart	price	avg_weekly_demand	demand_real	on_hand_begin	arrivals	sales_fulfilled	lost_sales	on_hand_end	orders_placed	inventory_position	reorder_point_s	order_up_to_S	lead_time_weeks	stockout_flag
2009-12-07 00:00:00	2.68	912.68	2,259.00	4,341	0	2,259	0	2,082	2,259	2,082	3,134	4,341	2	0
2009-12-14 00:00:00	2.66	912.68	3,003.00	2,082	0	2,082	921	0	2,082	2,259	3,134	4,341	2	1
2009-12-21 00:00:00	2.77	912.68	1,171.00	0	2,259	1,171	0	1,088	1,171	3,170	3,134	4,341	2	0
2009-12-28 00:00:00	4.50	1,611.25	12.00	1,088	2,082	12	0	3,158	5,072	4,329	6,951	9,401	2	0
2010-01-04 00:00:00	2.60	1,296.40	37.00	3,158	1,171	37	0	4,292	0	9,364	6,317	8,450	2	0
2010-01-11 00:00:00	2.63	1,375.33	1,770.00	4,292	5,072	1,770	0	7,594	620	7,594	6,089	8,214	2	0
2010-01-18 00:00:00	2.72	1,320.71	993.00	7,594	0	993	0	6,601	517	7,221	5,724	7,738	2	0
2010-01-25 00:00:00	2.70	1,339.75	1,473.00	6,601	620	1,473	0	5,748	1,250	6,265	5,534	7,515	2	0
2010-02-01 00:00:00	2.70	1,222.25	1,319.00	5,748	517	1,319	0	4,946	677	6,196	5,062	6,873	2	0
2010-02-08 00:00:00	2.74	950.62	830.00	4,946	1,250	830	0	5,366	0	6,043	3,611	4,946	2	0
2010-02-15 00:00:00	2.70	875.62	571.00	5,366	677	571	0	5,472	0	5,472	3,379	4,620	2	0
2010-02-22 00:00:00	2.91	946.50	579.00	5,472	0	579	0	4,893	0	4,893	3,051	4,257	2	0

Figure 26: Panel Overview Synthetic Inventory

Name	Type	Description
price	**FLOAT**	Weekly price carried over from the prep table.
avg_weekly_demand	**FLOAT**	Rolling average demand carried over.
demand_real	**FLOAT**	Historical weekly demand hitting the system (pre-stockout).
on_hand_begin	**INT**	Inventory at the start of the week (after arrivals, before sales).
arrivals	**INT**	Units that arrived this week from past orders (after LEAD_TIME_WEEKS).
sales_fulfilled	**INT**	Units actually sold (min of demand_real and available stock).
lost_sales	**INT**	Unmet demand due to stockout = demand_real − sales_fulfilled (≥0).
on_hand_end	**INT**	Inventory at week end after sales (before placing new order).
orders_placed	**INT**	Units ordered at week end (to raise position to s, or fixed Q).
inventory_position	**INT**	On-hand + on-order before placing new order (decision state).
reorder_point_s	**INT**	Weekly s used for decision (copied from prep table).
order_up_to_S	**INT**	Weekly S used for decision (copied from prep table).
lead_time_weeks	**INT**	The constant L used in the simulation.
stockout_flag	**BOOL**	1 if lost_sales > 0, else 0 (service failure indicator).

Figure 27: inventory_features_1

Name	Type	Description
SKU	**STR**	Which product (StockCode) to simulate. Filters the dataset
LEAD_TIME_WEEKS	**INT**	Supplier lead time L (weeks) between order placement and arrival.
SERVICE_LEVEL	**FLOAT**	Target cycle service level. Maps to a z-score used in safety stock
REVIEW_PERIOD	**INT**	Periodic review length T in weeks (weekly order-up-to policy).
ROLL_MEAN_W	**INT**	Window (weeks) for rolling average demand (baseline expectation)
MIN_PERIODS_MEAN	**INT**	Minimum weeks before rolling stats are computed
USE_WEIGHTED_PRICE	**BOOL**	If True (1), weekly price = revenue/qty (weighted); else simple mean
POLICY	**STR**	Inventory control: order-up-to (S,T) or reorder-point with fixed Q.
FIXED_Q_WEEKS	**INT**	For (s,Q): Q approx. Fixed_Q_Weeks * Avg_weekly_demand
Z_LOOKUP	**FLOAT**	z-score corresponding to Service_level (e.g. 95% = 1.6449).

Figure 28: inventory_features_2

Name	Type	Description
qty_week	**INT**	Sum of units sold in the week (historical)
price_mean	**FLOAT**	Simple weekly mean unit price
revenue	**FLOAT**	Weekly revenue = SUM(Qty *Price)
price	**FLOAT**	Weekly price used downstream (weighted if USE_WEIGHTED_PRICE)
demand_real	**FLOAT**	Historical weekly demand (acts as unconstrained demand for the sim)
avg_weekly_demand	**FLOAT**	Rolling mean over ROLL_MEAN_W (baseline demand expectation)
residual	**FLOAT**	Demand_real – avg_weekly_demand (proxy for forecast error)
std_weekly	**FLOAT**	Rolling std of residual (falls back to global std If 0).
mu_L	**FLOAT**	Mean demand over lead time: L*avg_weekly_demand
sigma_L	**FLOAT**	Std over lead time: std_weekly * sqrt(L)
mu_LplusT	**FLOAT**	Mean over L+T periods for order up-to computations
sigma_LplusT	**FLOAT**	Std over L+T periods
reorder_point_s	**INT**	"s" =muL + Z * sigmaL (time-varying reorder point)
order_up_to_S	**INT**	"S"= muL+T + Z * sigmaL+T (time-varying order-up-to).

Figure 29: inventory_features_3

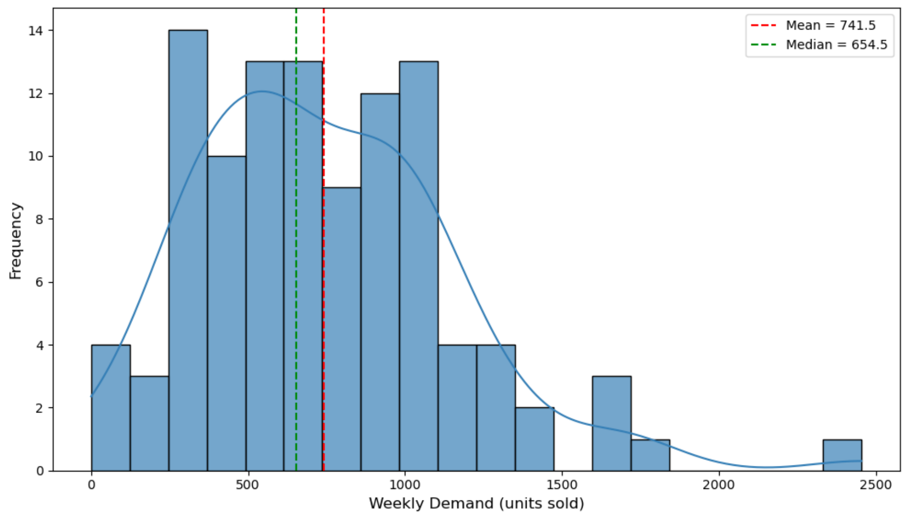

Figure 30: Histogram + KDE; Demand_real (SKU-week)

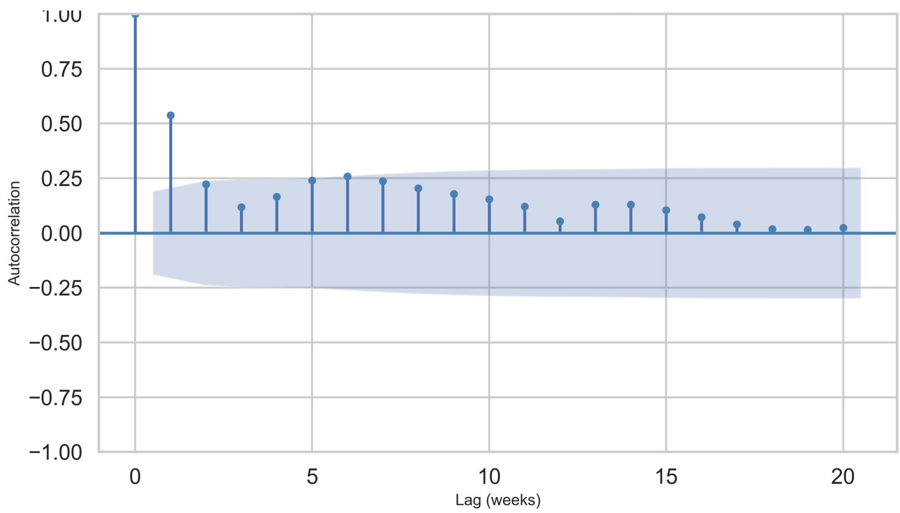

Figure 31: ACF of demand_real to show autocorrelation/seasonality

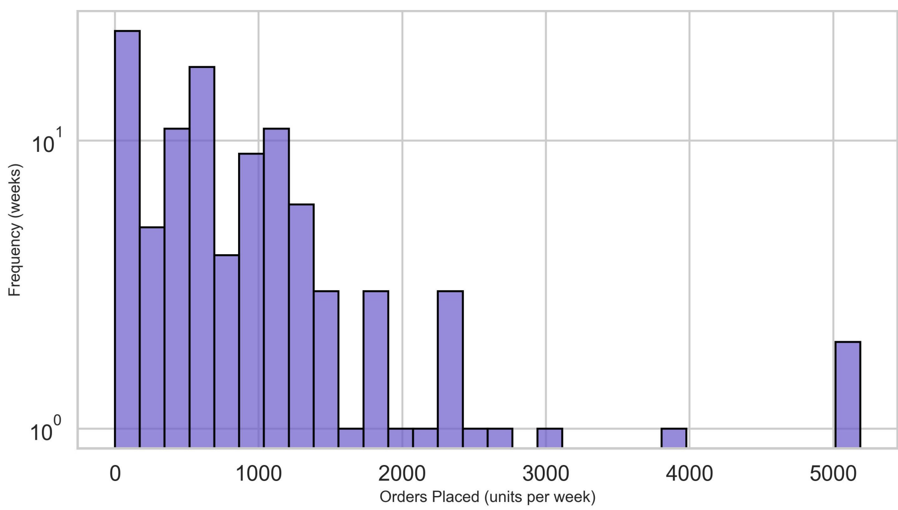

Figure 32: Histogram of orders_placed (often zero-inflated with occasional spikes).

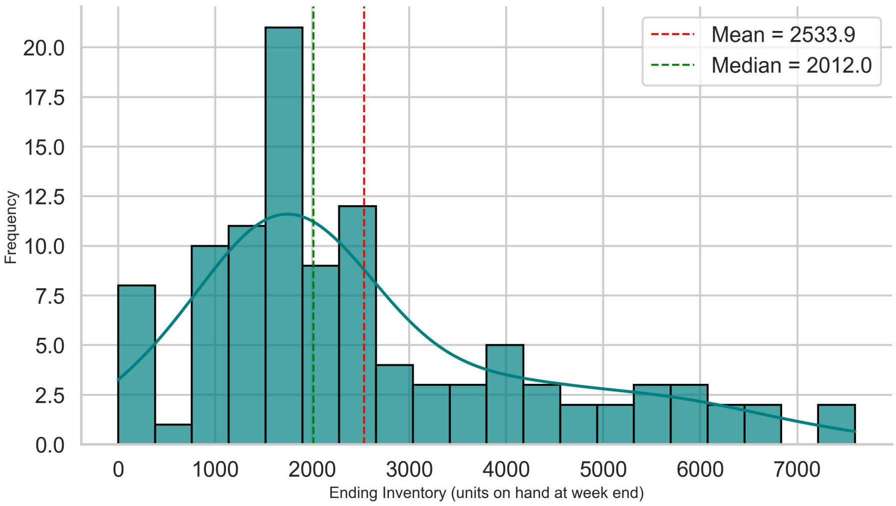

Figure 33: Histogram Ending Inventory.

Model	Time Index	Target Variable	Exogenous Features	Usage and Rationale
ETS	\`WeekStart\` (weekly)	\`demand_retail\`	None (pure univariate series: historical demand only).	Only past demand at weekly frequency to establish a simple, interpretable benchmark. Optional weekly seasonality (52) allows ETS to capture recurring patterns.
ARIMA	\`WeekStart\` (weekly)	\`demand_retail\`	None (pure univariate ARIMA(p,d,q) with constant; orders selected by AIC over a small grid).	Captures autocorrelation, differencing and short-term dynamics in the demand series, providing a stronger univariate baseline compared to ETS.
Ridge	\`WeekStart\` (weekly)	\`demand_retail\`	**Target lags:** \`lag_1\`–\`lag_4\` **Rolling stats of target:** \`roll_mean_4\`, \`roll_std_4\`, \`roll_mean_8\`, \`roll_std_8\` (on lagged series) **Calendar**: \`month\`, \`weekofyear\`, \`sin_woy\`, \`cos_woy\` **Exogenous**: \`log_price_wavg_retail\`, \`price_wavg_retail\`, \`price_mean_retail\`, \`on_hand_begin\`, \`on_hand_end\`, \`orders_placed\`, \`arrivals\`, \`inventory_position\`, \`reorder_point_s\`, \`order_up_to_S\`, \`lead_time_weeks\`.	Combines recent demand history, medium-term smoothing, seasonality proxies and operational/price covariates in a linear framework. This tests how far a regularised linear model with engineered features can improve upon purely univariate baselines.
XGBoost	\`WeekStart\` (weekly)	\`demand_retail\`	**Target lags:** \`lag_1\`–\`lag_12\` **Rolling stats of target**: \`roll_mean_4\`, \`roll_std_4\`, \`roll_mean_8\`, \`roll_std_8\`, \`roll_mean_12\`, \`roll_std_12\` (on lagged series) **Calendar**: \`weekofyear\`, \`sin_woy\`, \`cos_woy\` No contemporaneous exogenous variables.	Uses a richer set of lags and rolling statistics together with seasonal calendar encodings in a non-linear tree-based model. This allows XGBoost to capture complex, non-linear relationships in the demand dynamics beyond what Ridge can model.

Figure 34: Baseline Models Forecasting (Summary data inputs)

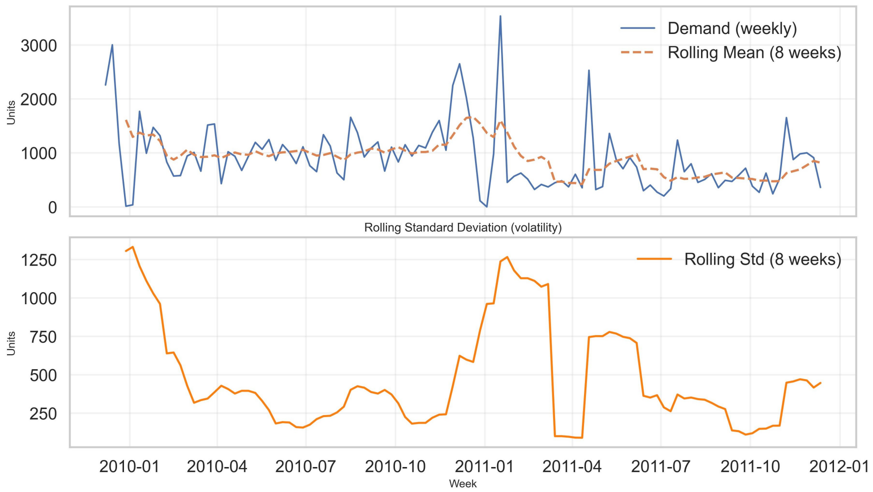

Figure 35: Rolling statistics: 8-week rolling mean and std of demand vs time (s, S)

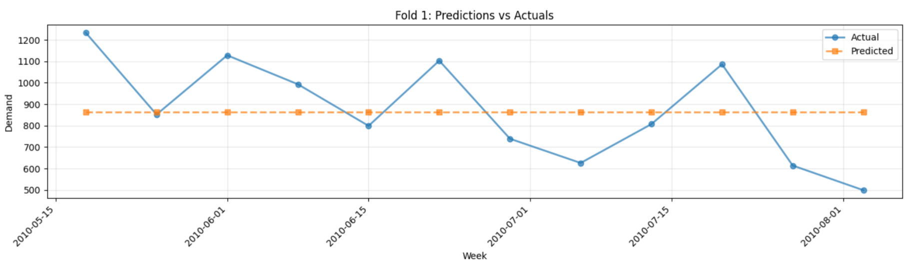

Figure 36: CV Time-Series: Fold 1

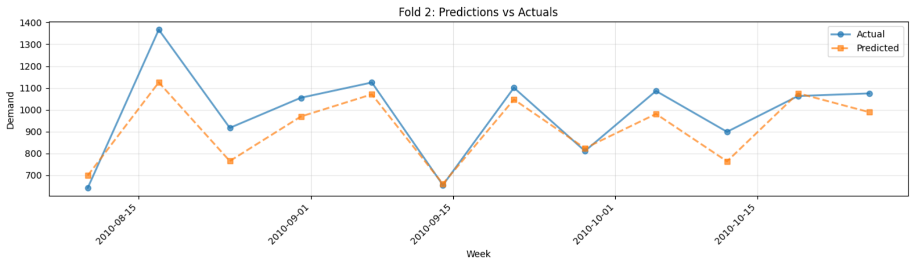

Figure 37: CV Time-Series: Fold 2

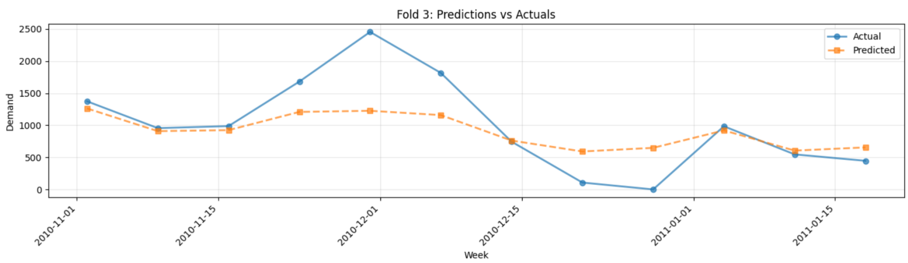

Figure 38: CV Time-Series: Fold 3

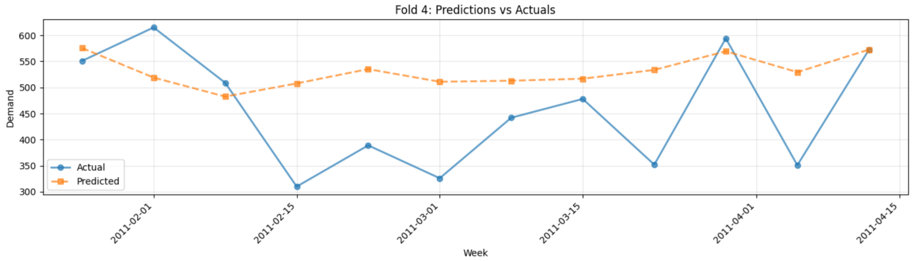

Figure 39: CV Time-Series: Fold 4

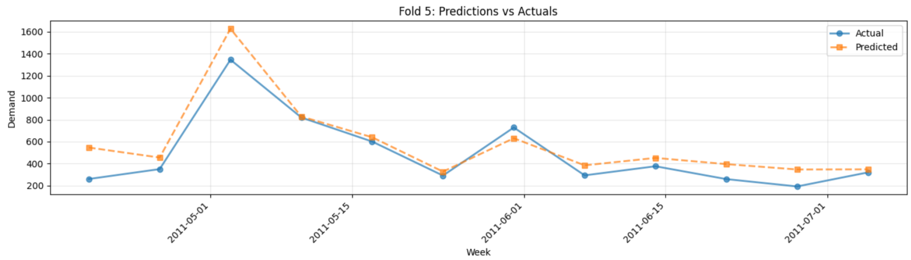

Figure 40: CV Time-Series: Fold 5

```python
# ---------------------- Context-aware feature builder -----------------------
def build_row_with_price_lags(
    history_df: pd.DataFrame,
    target_week: pd.Timestamp,
    p_candidate: float,
    feature_cols: List[str],
    TARGET: str,
    lags: List[int],
    roll_windows: List[int],
    date_col: str,
    price_col: str = "price",
) -> pd.Series:
    """
    Build a single feature row for week `target_week` by appending a row with price
    to the provided `history_df` (which must include past ACTUALS/PREV predictions
    """
    tmp = history_df.copy()
    tmp[date_col] = pd.to_datetime(tmp[date_col], errors="coerce")
    tmp = tmp.dropna(subset=[date_col]).sort_values(date_col)

    tw = pd.to_datetime(target_week)
    if (tmp[date_col] == tw).sum() == 0:
        tmp = pd.concat(
            [tmp, pd.DataFrame([{date_col: tw}])], ignore_index=True
        )

    tmp[price_col] = pd.to_numeric(tmp.get(price_col, np.nan), errors="coerce")
    tmp.loc[tmp[date_col] == tw, price_col] = float(p_candidate)

    # Lags/rolling of TARGET (leak-safe)
    for L in lags:
        tmp[f"lag_{L}"] = tmp[TARGET].shift(L)
    s = tmp[TARGET].shift(1)
    for W in roll_windows:
        tmp[f"roll_mean_{W}"] = s.rolling(W).mean()
        tmp[f"roll_std_{W}"] = s.rolling(W).std()

    # Seasonality
    woy = tmp[date_col].dt.isocalendar().week.astype(int)
    tmp["sin_woy"] = np.sin(2 * np.pi * woy / 52.0)
    tmp["cos_woy"] = np.cos(2 * np.pi * woy / 52.0)

    # Price lag 1 (needed by model)
    tmp["price_lag_1"] = tmp[price_col].shift(1)

    row = tmp.loc[tmp[date_col] == tw].copy().tail(1)
    row = row.reindex(columns=feature_cols, fill_value=0.0)
    return row.iloc[0]


def d_hat_at(
    ctx_df: pd.DataFrame, price_val: float, date_t: pd.Timestamp
) -> float:
    """Demand prediction for week `date_t` at price `price_val` using current conte
    row = build_row_with_price_lags(
        ctx_df,
        date_t,
        price_val,
        feature_cols,
        target,
        lags,
        rolls,
        date_col,
        "price",
    )
    X = (
        pd.DataFrame([row], columns=feature_cols)
        .apply(pd.to_numeric, errors="coerce")
        .fillna(0.0)
    )
    raw = float(reg.predict(X.values)[0])
    return max(0.0, CAL_FACTOR * raw)
```

Figure 41: Code: Weekly context feature builder for demand

```python
# --------------------- ROBUST PRICE CORRIDOR + WEEKLY RAMP -----------------
obs = pd.to_numeric(model_df["price"], errors="coerce").dropna().values
if obs.size == 0:
    raise RuntimeError("No historical prices to form a corridor.")

med = float(np.median(obs))
p10, p90 = float(np.quantile(obs, 0.10)), float(np.quantile(obs, 0.90))

# Trimmed corridor (drops outliers) and a symmetric band around the median
GLOBAL_BAND = 0.30  # ±30% around historical median
lo_trim = p10 * 0.95
hi_trim = p90 * 1.05
lo_band = med * (1.0 - GLOBAL_BAND)
hi_band = med * (1.0 + GLOBAL_BAND)

# Final global corridor (non-degenerate; ≥5% width)
PRICE_MIN = max(1e-6, max(lo_trim, lo_band))
PRICE_MAX = max(PRICE_MIN * 1.05, min(hi_trim, hi_band))

# Weekly ramp limit (max week-over-week move)
PRICE_RAMP_MAX = 0.08  # ≤ ±8% vs last executed price
LOCAL_BAND = 0.06  # default local search band ±6%


def clamp_price(p: float) -> float:
    return float(min(max(p, PRICE_MIN), PRICE_MAX))


print(
    f"[corridor] median={med:.4f} | global [{PRICE_MIN:.4f}, {PRICE_MAX:.4f}] | ram
)
```

Figure 42: Code: Price Corridor

```python
# --------------------- Capacity / safety helpers --------------------------
cap = int(MAX_WAREHOUSE_CAP)
safety_units = int(round(MAX_WAREHOUSE_CAP * SAFETY_STOCK_RATIO))
cycle_cap_nominal = cap - safety_units
safety_trigger_units = int(round(safety_units * SAFETY_USE_FRACTION))


def sellable_supply_for(inv_start: int, demand_int: int) -> int:
    cycle_allow = min(int(inv_start), cycle_cap_nominal)
    if int(demand_int) >= cycle_allow + safety_trigger_units:
        return min(int(inv_start), cap)
    return cycle_allow
```

Figure 43: Code: Warehouse capacity builder (CAP)

```python
def newsvendor_qstar(
    inv_start: int, d_hat: float, Co: float, Cu: float
) -> int:
    q_max = max(5, int(math.ceil(2.0 * (float(d_hat) + 1e-9))))
    q = np.arange(0, q_max + 1, dtype=int)
    over = np.maximum(inv_start + q - d_hat, 0.0)
    under = np.maximum(d_hat - (inv_start + q), 0.0)
    cost = Co * over + Cu * under
    return int(q[int(np.argmin(cost))])
```

Figure 44: Code: Newsvendor function builder

```python
def sell_through_price(
    ctx_df: pd.DataFrame,
    date_t: pd.Timestamp,
    supply_target: int,
    pmin: float = None,
    pmax: float = None,
) -> float:
    """Find a price in [pmin,pmax] that sells approximately `supply_target` units v
    pmin = PRICE_MIN if pmin is None else float(pmin)
    pmax = PRICE_MAX if pmax is None else float(pmax)
    pmin, pmax = clamp_price(pmin), clamp_price(pmax)
    if pmin >= pmax:
        return float(pmin)

    d_lo = d_hat_at(ctx_df, pmin, date_t)
    d_hi = d_hat_at(ctx_df, pmax, date_t)

    # If no crossing, choose nearest bound that best meets the target.
    if d_lo < supply_target and d_hi < supply_target:
        return float(pmin)  # cannot sell all even at low price
    if d_lo > supply_target and d_hi > supply_target:
        return float(pmax)  # demand too high, push to max price

    lo, hi = float(pmin), float(pmax)
    for _ in range(32):
        mid = 0.5 * (lo + hi)
        if d_hat_at(ctx_df, mid, date_t) > supply_target:
            lo = mid
        else:
            hi = mid
    return float(0.5 * (lo + hi))
```

Figure 45: Code: Sell-through bisection builder

```python
    # expected over/under cost at t+2 with chosen q*
    over = max(inv_before_new_t2 + q_star - d2, 0.0)
    under = max(d2 - (inv_before_new_t2 + q_star), 0.0)
    future_cost_t2 = Co * over + Cu_eff * under

    # joint objective (small smooth term around last price)
    joint_obj = (
        profit_now
        - FUTURE_COST_WEIGHT * future_cost_t2
        - PRICE_SMOOTH * (p - p_prev) ** 2
```

Figure 46: Code: Future cost builder (FOC)

```
# ---------------------- JOINT LOOP ----------------------------------------
for t in range(horizon_weeks):
    date_t = last_date + pd.Timedelta(days=7 * (t + 1))
    date_t1 = last_date + pd.Timedelta(days=7 * (t + 2))
    date_t2 = last_date + pd.Timedelta(days=7 * (t + 3))

    # receipts for this week
    received_now = pipeline.popleft() if len(pipeline) > 0 else 0
    inv = min(cap, inv + int(received_now))

    # commit order (arrives in t+2)
    pipeline.append(int(q_star))

    # update context with chosen p* and predicted demand (for lags/price_lag_1)
    context = pd.concat(
        [
            context,
            pd.DataFrame(
                [{date_col: date_t, target: d_hat_pstar, "price": p_star}]
            ),
        ],
        ignore_index=True,
    )

    # advance inventory to next week
    inv = int(leftover_units)
```

Figure 47: Code: Joint Policy builder

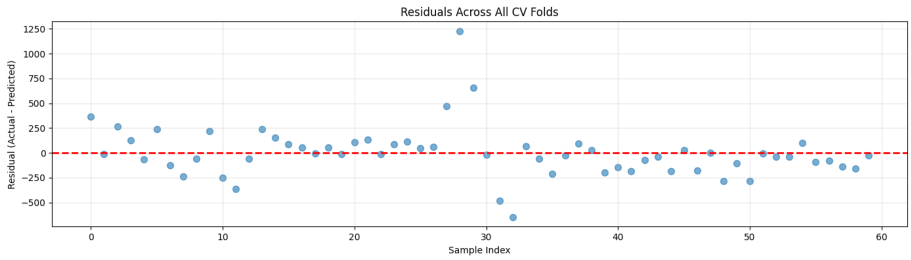

Figure 48: Residuals across all folds

Component	Version
CPU	1.3 GHz Dual Core Intel Core i5
RAM	4GB
Storage	500GB SSD
OS	Mac OS “Big Sur” 11.7.10
Python	3.12.7
XGBoost	3.0.0
Numpy	1.26.4
Pandas	2.2.2

Figure 49: System Components List